THE MINISTRY OF INFORMATION DEVELOPMENTTECHNOLOGY AND COMMUNICATIONSTHE REPUBLIC OF UZBEKISTAN
TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

Permission to defense

Head of TI department

Eshmuradov A.M. _____

«____»_____2015.

DEVELOPING THE MODEL FOR MEASURING AND EVALUATING OF PARAMETERS OF THE AUTONOMOUS SOLAR POWER STATION

diploma thesis

Graduate	_____ signature	Begmatov SH. A. S.N.M.
Supervisor	_____signature	Isaev R. I. S.N.M.
SVA& ecology	_____signature	Qodirov F.M. S.N.M.
Review	_____signature	Utabayev B. S. S.N.M.

TASHKENT – 2015

THE MINISTRY OF INFORMATION DEVELOPMENTTECHNOLOGY AND COMMUNICATIONSTHE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

Faculty Telecommunication Technologies Department TE

Specialization 5351300 –Telecommunication

CONFIRM

Head of department _____

«____»_____2015.

Graduate Begmatov Shohruh Abduvaxob o'g'li

N.S.M.

Developing the model for measuring and evaluating of parameters of the autonomous solar power station

A S S I G N M E N T

1. **Diploma thesis confirmed 2014.25.12 according to the order № 1467**
2. **Deadline to give diploma thesis before defense:** *01.06.15 y.*
3. **Previous information connected with diploma:** *Taken information from Practical training, technical manual, journal and internet resources*
4. **Measuring:** 1.Concepts and measuring parameters of autonomous solar power system, 2. Measuring parameters of autonomous solar power system, 3. Model of the Monitoring system for autonomous solar power plants, 4. Safety of vital activity and ecology.
5. **Listed of graphic materials:** diploma thesis presentation
6. **Assignment taken data:** *12.02.15*

Supervisor _____

Signature

Graduate _____

Signature

7. Advices of sub chapters of the work:

Chapter name	Advisor	Signature, data	
		Given assignment	Taken assignment
1,2,3 chapters	Isaev R. I.	16.02.15	16.02.15
4 chapter	Qodirov F. M.	17.04.15	17.04.15

8. Order to be accomplished of the work:

Or.	Name of chapters	Performance data	Supervisor
1.	Concepts and measuring parameters of autonomous solar power system	28.03.15y.	
2.	Measuring parameters of autonomous solar power system	26.04.15y.	
3.	Model of the Monitoring system for autonomous solar power plants	12.05.15y.	
4.	Safety of vital activity and ecology	28.05.15y	

Graduate _____ 2015 year « »

signature

Supervisor _____ 2015 year « »

signature

Ushbu bitiruv malakaviy ish avtonom quyosh elektrostansiyasi parametrlarini o’lchash va baxolash modelini ishlab chiqishga qaratilgan. Ushbu bitiruv malakaviy ishda avtanom quyosh elektrostansiyasi parametrlarini va avtonom uyosh elektronstasiyasi holatini monitoring qilish haqida yoritib o’tilgan. Avtonom quyosh elektrostansiyalarining monitoring tizimi fizik va dasturiy jihatdan yaratilgan.

Hamda hayot faoliyati xafsizligi va ekologiya masalalari ko’rib chiqilgan.

Данная квалификационная работа направлена на разработку модели измерения и оценки параметров автономных солнечных электростанций. В этой квалификационной работе подробно описаны параметры работы автономных солнечных электростанций и принципы мониторинга состояния солнечных электростанций. Разработаны программные и аппаратные средства мониторинга автономней солнечной электростанции.

Рассмотрены вопросы касательно безопасной жизни деятельности и экологические проблемы.

This qualification work is aimed at developing the model for measuring and evaluating of parameters of the autonomous solar power station. In this qualifying work described in detail the parameters of the autonomous solar power and principles of monitoring the state of solar power plants. Developed software and hardware of monitoring of the autonomous solar power station.

Addressed issues relating to the safety of life activity and environmental problems.

CONTENTS

INTRODUCTION

As the President of our country Islam Karimov marked, at the council of cabinet Ministers devoted to totals of social and economic development in 2014 and the major priority directions of the economic program for 2015, - "implementation of measures for development of electronic commerce has importance. Now in the country over 10 million plastic cards are let out, including about 2,5 million are online plastic cards. Already today it is possible to pay via the Internet telecommunication and utilities. It is necessary to remove obstacles in the shortest periods and it is essential to broaden this sphere" [1].

In Uzbekistan, for many decades, attention has been paid to the solar energy researches. The first development in the field of solar energy in Uzbekistan began nearly 80 years ago, in the past 1932, when a laboratory of solar engineering was created in Samarkand. During 1940 and 1950s the first parabolic concentrator was built and the scientific investigations were commenced on solar energy conversion into electricity, after that the first photovoltaic installation was built.

Figure 1. Map of Uzbekistan

In 1943, The physical-technical institute (PhTI) was organized, which was the first academic institution engaged in extensive research in the field of physics and engineering in a territory of Central Asia.

In 1986, the Institute of Physics-Sun was organized on the basis of The physical-technical institute. In 1987 a unique optical mirror complex with a big solar furnace, which thermal power is 1 MW was put into operation.

Since 1965, The physical-technical Institute of Physics-Sun has been publishing The international journal "GELIOTEHNIKA". The journal is translated into English by the company "Allerton Press" (USA) and it is published in the United States under the name "Applied Solar Energy".

The territory of Uzbekistan is 447.700 km^2 [Fig. 1]. More than 70% of its surface is suitable to build and install solar power stations. It means we have more requirement lands for using type of this energy. In Fig. 2 shown, the gross potential of solar radiation in the Republic of Uzbekistan is estimated to be in the range from 525 billion kWh up to 760 billion kWh [2].

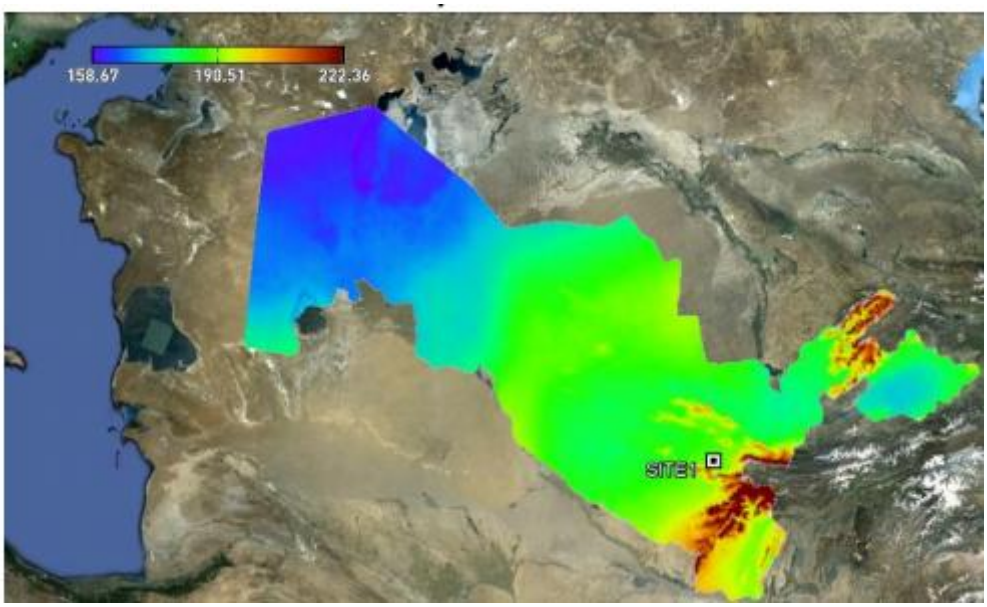

Figure 2. The gross potential of solar radiation in the Republic of Uzbekistan

In 1987, under the guidance of prof. Isaev R.I. It was first developed hybrid solar-wind power supply system on the power of 250 W, which is more than 10

years, was tested on the ground in the foothills of the (educational-production base TUIT) and showed a high level of reliability and performance. In 1998, on the basis of these data, the grant of the European Commission in the framework of the Inco-Copernicus in the foothills of the Tashkent region (Repeater "Charvak") built and put into trial operation in August 2000, Central Asia's first hybrid solar-wind power 5 kW with batteries with a total capacity of 1520 A / h.

Telecommunications networks in Uzbekistan are largely based on Soviet-built infrastructure but with many modern additions, making the country one of the leading influences in the region in informational development.

The role of solar and wind power plants in the field of telecommunications

"Uzbektelecom" actively uses solar and wind generators at the facilities of communication May 5, 2015 "Uzbektelecom" in 2012, implemented a number of projects for the installation of alternative energy sources such as solar and wind generators to ensure uninterrupted supply of communication facilities and power-saving.

The Decree of the President of the Republic Uzbekistan № UP-4512 "On measures for further development of alternative energy sources" on March 1, 2013, set specific targets for the introduction of solar energy in practice with international experience.

The report of the President Islam Karimov at the VI International Forum of the Asia Solar Energy, held November 22, 2013 in Tashkent emphasized the efficiency of the use of solar energy and prospects for the development of solar energy in Uzbekistan.

"Uzbektelecom" as a national telecom operator is a major consumer of electricity, as the communication objects require uninterrupted and reliable power supply around the clock.

To save energy and improve reliability in power supply in the years 2012-2014 in the departments of "Uzbektelecom", together with the business entities have been implemented a pilot project to equip remote and difficult communication objects solar generators. For example, in 2012 a branch

UZMOBILE in 2 villages of Surkhandarya region in 2013 in 3 mountain village of Tashkent region, 2013-2014 rural ATS 12 remote areas and BTS (base stations) and PPC (radio relay stations) 44-remote areas were constructed solar power plants.

The effectiveness of the project in the first place to the population including seamless communication mobile, Internet, IPTV and other services.

Thus, for example, installed solar power branch UZMOBILE on BPS in the village "Duoba" Jizzakh region provides 8 villages uninterrupted communication.

In the field of alternative energy JSC "Uzbektelecom" and cooperates with international organizations. According to the order of CIA Uz from April 1, 2011 №112 between JSC "Uzbektelecom" and the International Telecommunication Union treaty was signed on September 7, 2011. the project "uninterrupted power supply of telecommunication facilities in villages in remote areas." A branch of "TTT" on site "PPC Sambar" made installation of devices to ensure stable sources of energy (electricity, solar, wind, or hybrid). In June of 2015 under the guidance of prof. Isaev R.I. will be commissioned in solar and wind power.

To perform the planned works on the installation of solar power plants in the telecommunication networks of the company, in all regions of the country are actively involved students of professional colleges.

The business plan of JSC "Uzbektelecom" in 2015 planned to build 92 solar power plants at the facilities of communication and exchanges in remote areas and cities for a stable electricity supply.

The introduction of solar and wind generators at the facilities of communication not only provides a direct economic benefit of saving electricity and diesel fuel. Even more significant economic effect is achieved due to the fact that the communication objects require uninterrupted and reliable power supply around the clock, both for the population and state organizations and business entities. After all, the main task of communicators to provide reliable and high-quality communication, 365 days a year, 24 hours a day.

1. CONCEPTS AND MEASURING PARAMETERS OF AUTONOMOUS SOLAR POWER SYSTEM

1.1. Solar energy and photovoltaic technology

Solar energy is natural radiant light and heat from the sun harnessed using a range of ever-evolving technologies such as solar heating, photovoltaics, solar thermal energy, solar architecture and artificial photosynthesis.

It is an important source of renewable energy and its technologies are broadly characterized as either passive solar or active solar depending on the way they receiving and distribute solar energy or convert it into solar power. Active solar techniques include the use of photovoltaic systems, concentrated solar power and solar water heating to harness the energy. Passive solar techniques include basic orienting a building to the Sun, selecting materials with favorable thermal mass or light dispersing properties, and designing spaces that naturally circulate air.

Energy comes from the Sun and the Earth receives 174,000 terawatts (TW) of incoming solar radiation (insolation) at the upper atmosphere. Approximately 30% is reflected back to space while the rest is absorbed by clouds, oceans and land masses. The light spectrum of solar at the Earth's surface is mostly spread across the visible and near-infrared ranges with a small part in the near-ultraviolet. Most people around the world live in areas with insolation levels of 150 to 300 watt per square meter or 3.5 to 7.0kWh/m^2 per day.

Earth's land surface, atmosphere and oceans absorb solar radiation, and this raises their temperature. Warm air containing evaporated water from the oceans rises, causing atmospheric circulation or convection. When the air reaches a high altitude, where the temperature is low, water vapor condenses into clouds, which rain onto the Earth's surface, completing the water cycle. The latent heat of water condensation amplifies convection, producing atmospheric phenomena such as wind, cyclones and anti-cyclones. Sunlight absorbed by the oceans and land

masses keeps the surface at an average temperature of 14 °C. By using photosynthesis green plants convert solar energy into chemical energy, which produces food, wood and the biomass from which fossil fuels are derived.

Solar power is the conversion of sunlight into electrical form, either directly using photovoltaics [PV – Fig. 1.1], or indirectly using concentrated solar power (CSP). Concentrated solar power systems use mirrors or lenses and tracking systems to focus a large area of sunlight into a small beam. Photovoltaics convert light into electrical form using the photovoltaic effect.

Figure 1.1. Photovoltaic stations

Photovoltaics were initially, and still are, used to power small and medium-sized applications, from the calculator powered by a single solar cell to off-grid homes powered by a photovoltaic array. They are an important and relatively inexpensive source of electrical energy where grid power is inconvenient, unreasonably expensive to connect, or simply unavailable. However, as the cost of solar electricity is falling, solar power is also increasingly being used even in grid-connected situations as a way to feed low-carbon energy into the grid.

Photovoltaic power systems A solar cell, or photovoltaic cell (PV), is a device that converts light into electric current using the photovoltaic effect. The first solar cell was constructed by Charles Fritts in the 1880s. The German industrialist Ernst Werner von Siemens was among those who recognized the importance of this discovery. In 1931, the German engineer Bruno Lange developed a photo cell using silver selenite in place of copper oxide, although the prototype selenium cells converted less than 1% of incident light into electricity. Following the work of Russell Ohl in the 1940s, researchers Gerald Pearson, Calvin Fuller and Daryl Chapin created the silicon solar cell in 1954. These early solar cells cost 286 USD/watt and reached efficiencies of 4.5–6%.

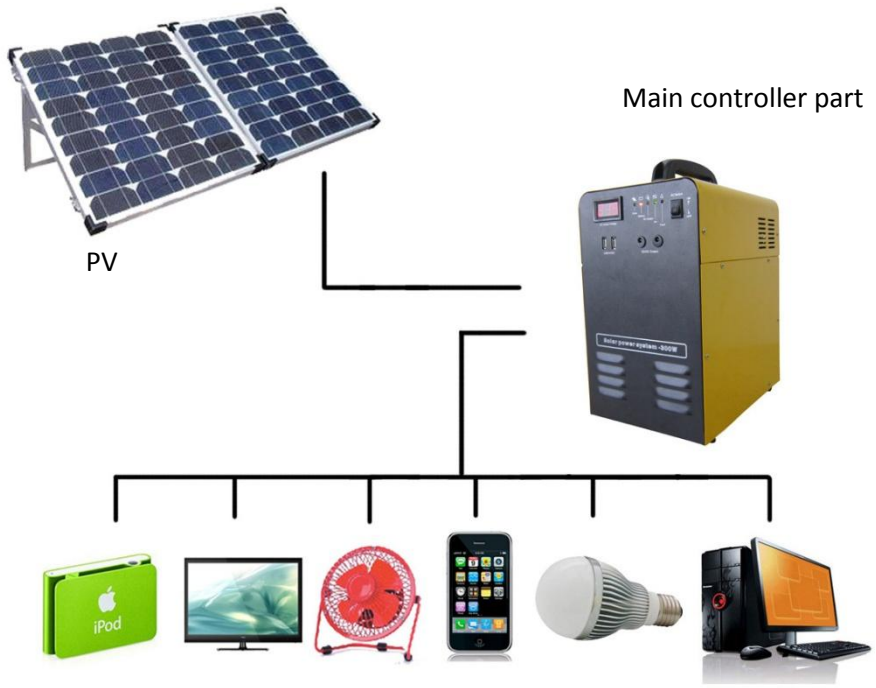

Figure 1.2. PV power system

The array of a photovoltaic power system [Fig. 1.2], or PV system, produces direct current (DC) power which fluctuates with the sunlight's intensity. For practical use this usually requires conversion to certain desired voltages or alternating current (AC), through the use of inverters . Multiple solar cells are connected inside modules [7]. Modules are wired together to form arrays, then tied

to an inverter, which produces power at the desired voltage, and for AC, the desired frequency/phase.

Many residential PV systems are connected to the grid wherever available, especially in developed countries with large markets. In these grid-connected PV systems, use of energy storage is optional. In certain applications such as satellites, lighthouses, or in developing countries, batteries or additional power generators are often added as back-ups. Such stand-alone power systems permit operations at night and at other times of limited sunlight.

There are also many large plants under construction. The Desert Sunlight Solar Farm is a 550 MW power plant under construction in Riverside County, California, that will use thin-film CdTe-modules made by First Solar. The Blythe Solar Power Project is a 485 MW project under construction also in California. As of November 2014, the 550-megawatt Topaz Solar Farm is the largest photovoltaic power plant in the world.

Commercial concentrating solar thermal power (CSP) plants were first developed in the 1980s. The 377 MW Ivanpah Solar Power Facility, located in California's Mojave Desert, is the world’s largest solar thermal power plant project. Other large CSP plants include the Solnova Solar Power Station (150 MW), the Andasol solar power station (150 MW), and Extresol Solar Power Station (150 MW), all in Spain. The principal advantage of CSP is the ability to efficiently add thermal storage, allowing the dispatching of electricity over up to a 24-hour period. Since peak electricity demand typically occurs at about 5 pm, many CSP power plants use 3 to 5 hours of thermal storage.

1.2. Advantages of autonomous solar power system

Renewable energy sources in general, and Solar Energy source in particular, has the potential to provide energy services with zero or almost zero emission. The solar energy is abundant and no other source in renewable energy is like solar energy. Every technology has its own advantages and disadvantages. As the solar

insolation and atmospheric conditions vary significantly from place to place, efficiency of solar energy also differs accordingly.

Global Warming is the biggest issue facing the future of our planet. We all need to act now by reducing our greenhouse gas emissions and lead the way for our children and the local community in the acceptance and installation of renewable energy technologies.

Solar Energy helps reduce your carbon footprint as it is a clean energy alternative. Solar Energy does not contribute to global warming, acid rain or smog. Solar Power therefore, actively contributes to the decrease of harmful greenhouse gas emissions.

Less pollution in the air means cleaner air for us to breathe, therefore increasing our life expectancy.

Solar energy is currently only a small percentage of our total energy use. You are helping Australia to achieve its 15% green energy target by 2020 and global targets of 20% at the same time by switching to Solar power.

Solar energy produces no waste and no pollution. Therefore limiting your impact on climate change and global warming.

Solar Power is Quiet. In fact, solar panels are silent. They convert sun light into electricity without making a sound.

So we have any advantages from solar power plants. Firstly, it is very economically and more safety than other electrical power station. Secondly, this project can help to protect our environment of Uzbekistan. That is way we can say it is very ecological. Finally, it can work autonomously.

Some advantages have followed for example:

– adds value to your property;
– reduces greenhouse gas emissions;
– better for the environment and future generations;
– clean green silent energy production;
– long lasting with system life expectancies in excess of 25 years;
– extensive warranties of up to 25 years;

– clean, silent energy production;
– suitable for a variety of locations and building types;
– reduce Uzbekistan's dependence on coal , oil and (uranium);
– clean, green silent energy production;
– solar credit rebates that are available to everyone, including businesses.

1.3. Components of an autonomous solar power system

Using solar power to produce electricity is not the same as using solar to produce heat. Solar thermal principles are applied to produce hot fluids or air. Photovoltaic principles are used to produce electricity. A solar panel (PV panel) is made of the natural element, silicon, which becomes charged electrically when subjected to sun light. Solar panels are directed at solar south in the northern hemisphere and solar north in the southern hemisphere (these are slightly different than magnetic compass north-south directions) at an angle dictated by the geographic location and latitude of where they are to be installed. Typically, the angle of the solar array is set within a range of between site-latitude-plus 15 degrees and site-latitude-minus 15 degrees, depending on whether a slight winter or summer bias is desirable in the system.

Many solar arrays are placed at an angle equal to the site latitude with no bias for seasonal periods. This electrical charge is consolidated in the PV panel and directed to the output terminals to produce low voltage (Direct Current) - usually 6 to 24 volts. The most common output is intended for nominal 12 volts, with an effective output usually up to 17 volts. A 12 volt nominal output is the reference voltage, but the operating voltage can be 17 volts or higher much like your car alternator charges your 12 volt battery at well over 12 volts. So there's a difference between the reference voltage and the actual operating voltage. The intensity of the Sun's radiation changes with the hour of the day, time of the year and weather conditions. To be able to make calculations in planning a system, the total amount of solar radiation energy is expressed in hours of full sunlight per m², or Peak Sun

Hours. This term, Peak Sun Hours, represents the average amount of sun available per day throughout the year. It is presumed that at "peak sun", 1000 W/m²of power reaches the surface of the earth. One hour of full sun provides 1000 Wh per m² = 1 kWh/m² - representing the solar energy received in one hour on a cloudless summer day on a one-square meter surface directed towards the sun. To put this in some other perspective, the United States Department of Energy indicates the amount of solar energy that hits the surface of the earth every +/- hour is greater than the total amount of energy that the entire human population requires in a year. Another perspective is that roughly 100 square miles of solar panels placed in the southwestern U.S. could power the country. The daily average of Peak Sun Hours, based on either full year statistics, or average worst month of the year statistics, for example, is used for calculation purposes in the design of the system. To see the average Peak Sun Hours for your area consult the solar maps in the solar calculator section of the Sun force website. So it can be concluded that the power of a system varies, depending on the intended geographical location. Folks in the northeastern U.S. will need more solar panels in their system to produce the same overall power as those living in Arizona. Sun forces technical support representatives can advise you on this if you have any doubts about your area.

Components used to provide solar power:

The four primary components for producing electricity using solar power, which provides common 110-120 volt AC power for daily use are: Solar panels, charge controller, battery and inverter. Solar panels charge the battery, and the charge regulator insures proper charging of the battery. The battery provides DC voltage to the inverter, and the inverter converts the DC voltage to normal AC voltage. If 220-240 volts AC is needed, then either a transformer is added or two identical inverters are series-stacked to produce the 240 volts.

Solar Panels

Pointed towards the sun, solar panels capture the energy in sunlight and main part of can convert it directly to DC electricity. There are three general

families of solar panels on the market today – single crystal silicon, polycrystalline silicon, and thin film.

PV modules are very durable and, because there are no moving parts, long-lasting. Most carry 25 year warranties. Solar panels are assigned a rating in watts based on the maximum power they can produce under ideal sun and temperature conditions per hour. You can use the rated output to help determine how many voltages you need. Multiple modules mounted together are called an array.

Solar panels generate free power from the sun by converting sunlight to electricity with no moving parts, zero emissions, and no maintenance. The solar panel, the first component of a electric solar energy system, is a collection of individual silicon cells that generate electrical current from sunlight. The photons (light particles) produce an electrical current as they strike the surface of the thin silicon wafers. A single solar cell produces only about 1/2 (.5) of a volt. However, a typical 12 volt panel about 25 inches by 54 inches will contain 36 cells wired in series to produce about 17 volts peak output. If the solar panel can be configured for 24 volt output, there will be 72 cells so the two 12 volt groups of 36 each can be wired in series, usually with a jumper, allowing the solar panel to output 24 volts. When under load (charging batteries for example), this voltage drops to 12 to 14 volts (for a 12 volt configuration) resulting in 75 to 100 watts for a panel of this size.

Multiple solar panels can be wired in parallel to increase current capacity (more power) and wired in series to increase voltage for 24 or even higher voltage systems. The advantage of using a higher voltage output at the solar panels is that smaller wire sizes can be used to transfer the electric power from the solar panel array to the charge controller & batteries. Since copper has gone up considerably in the last few years, purchasing large copper wiring and cables is quite expensive.

The 3 basic types of Solar Panels

Monocrystalline solar panels: The most efficient and expensive solar panels are made with Monocrystalline cells. These solar cells use very pure silicon and involve a complicated crystal growth process. Long silicon rods are produced

which are cut into slices of .2 to .4 mm thick discs or wafers which are then processed into individual cells that are wired together in the solar panel.

Polycrystalline solar panels: Often called Multi-crystalline, solar panels made with Polycrystalline cells are a little less expensive & slightly less efficient than Monocrystalline cells because the cells are not grown in single crystals but in a large block of many crystals. This is what gives them that striking shattered glass appearance. Like Monocrystalline cells, they are also then sliced into wafers to produce the individual cells that make up the solar panel.

Amorphous solar panels: These are not really crystals, but a thin layer of silicon deposited on a base material such as metal or glass to create the solar panel. These Amorphous solar panels are much cheaper, but their energy efficiency is also much less so more square footage is required to produce the same amount of power as the Monocrystalline or Polycrystalline type of solar panel. Amorphous solar panels can even be made into long sheets of roofing material to cover large areas of a south facing roof surface.

The output of a solar panel is usually stated in watts, and the wattage is determined by multiplying the rated voltage by the rated amperage. The formula for wattage is VOLTS times AMPS equals WATTS. So for example, a 12 volt 60 watt solar panel measuring about 20 X 44 inches has a rated voltage of 17.1 and a rated 3.5 amperage. V x A = W 17.1 volts times 3.5 amps equals 60 watts If an average of 6 hours of peak sun per day is available in an area, then the above solar panel can produce an average of 360 watt hours of power per day; 60w times 6 hrs. = 360 watt-hours. Since the intensity of sunlight contacting the solar panel varies throughout the day, we use the term "peak sun hours" as a method to smooth out the variations into a daily average. Early morning and late-in-the-day sunlight produces less power than the mid-day sun. Naturally, cloudy days will produce less power than bright sunny days as well. When planning a system your geographical area is rated in average peak sun hours per day based on yearly sun data. Average peak sun hours for various geographical areas is listed in the above section. Solar panels can be wired in series or in parallel to increase voltage or amperage

respectively, and they can be wired both in series and in parallel to increase both volts and amps.

Series wiring refers to connecting the positive terminal of one panel to the negative terminal of another. The resulting outer positive and negative terminals will produce voltage the sum of the two panels, but the amperage stays the same as one panel. So two 12 volt / 3.5 amp panels wired in series produces 24 volts at 3.5 amps. Four of these wired in series would produce 48 volts at 3.5 amps. Parallel wiring refers to connecting positive terminals to positive terminals and negative to negative. The result is that voltage stays the same, but amperage becomes the sum of the number of panels. So two 12 volt / 3.5 amp panels wired in parallel would produce 12 volts at 7 amps. Four panels would produce 12 volts at 14 amps. Series parallel wiring refers to doing both of the above - increasing volts and amps to achieve the desired voltage as in 24 or 48 volt systems. The following diagram reflects this. In addition, the four panels below can then be wired in parallel to another four and so on to make a larger array.

The Charge Controller

The Charge Controller is installed between the Solar Panel array and the Batteries where it automatically maintains the charge on the batteries using the 3 stage charge cycle just described [Fig. 1.3].

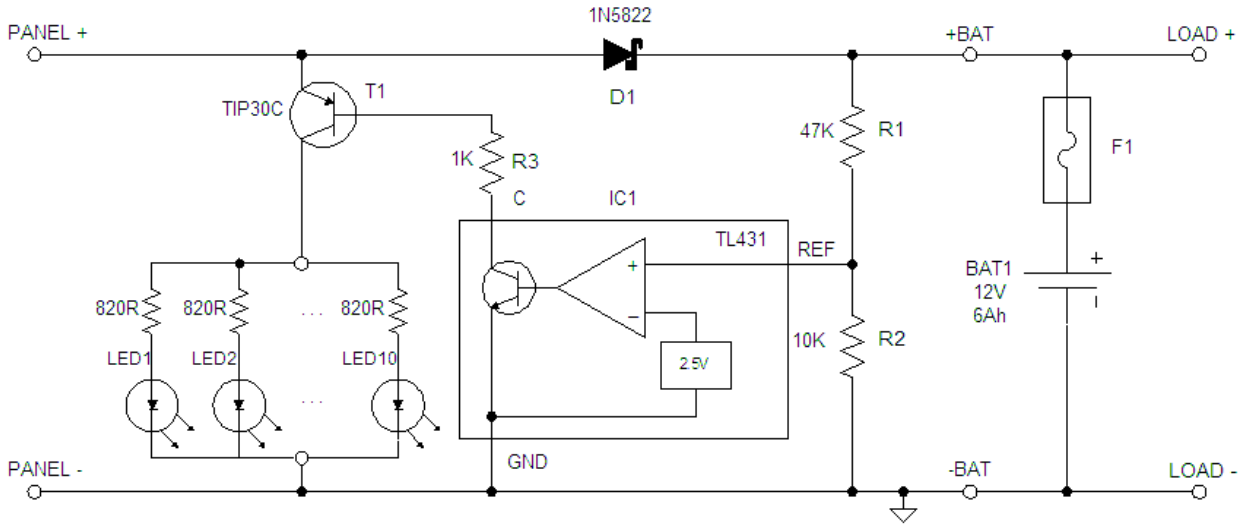

Figure 1.3. Example for charge controller

The Power Inverter can also charge the batteries if it is connected to the AC utility grid or in the case of a standalone system, your own AC Generator

Why a Charge Controller is necessary

Since the brighter the sunlight, the more voltage the solar cells produce, the excessive voltage could damage the batteries. A charge controller is used to maintain the proper charging voltage on the batteries. As the input voltage from the solar array rises, the charge controller regulates the charge to the batteries preventing any overcharging.

Modern multi-stage charge controllers

Most quality charge controller units have what is known as a 3 stage charge cycle that goes like this :

1) BULK : During the Bulk phase of the charge cycle, the voltage gradually rises to the Bulk level (usually 14.4 to 14.6 volts) while the batteries draw maximum current. When Bulk level voltage is reached the absorption stage begins.

2) ABSORPTION : During this phase the voltage is maintained at Bulk voltage level for a specified time (usually an hour) while the current gradually tapers off as the batteries charge up.

3) FLOAT : After the absorption time passes the voltage is lowered to float level (usually 13.4 to 13.7 volts) and the batteries draw a small maintenance current until the next cycle.

The relationship between the current and the voltage during the 3 phases of the charge cycle can be shown visually by the graph below [13].

Battery

The Deep Cycle batteries used are designed to be discharged and then re-charged hundreds or thousands of times. These batteries are rated in Amp Hours (ah) usually at 20 hours and 100 hours. Simply stated, amp hours refers to the amount of current - in amps - which can be supplied by the battery over the period of hours.

For example, a 350ah battery could supply17.5 continuous amps over 20 hours or 35 continuous amps for 10 hours. To quickly express the total watts

potentially available in a 6 volt 360ah battery; 360ah times the nominal 6 volts equals 2160 watts or 2.16kWh (kilowatt-hours). Like solar panels, batteries are wired in series and or parallel to increase voltage to the desired level and increase amp hours. The battery should have sufficient amp hour capacity to supply needed power during the longest expected period "no sun" or extremely cloudy conditions. A lead-acid battery should be sized at least 20% larger than this amount. If there is a source of back-up power, such as a standby generator along with a battery charger, the battery bank does not have to be sized for worst case weather conditions. The size of the battery bank required will depend on the storage capacity required, the maximum discharge rate, the maximum charge rate, and the minimum temperature at which the batteries will be used. During planning, all of these factors are looked at, and the one requiring the largest capacity will dictate the battery size. One of the biggest mistakes made by those just starting out is not understanding the relationship between amps and amp-hour requirements of 120 volt AC items versus the effects on their DC low voltage batteries.

For example, say you have a 24 volt nominal system and an inverter powering a load of 3 amps, 120VAC, which has a duty cycle of 4 hours per day. You would have a 12 amp hour load (3A X 4 hrs = 12ah). However, in order to determine the true drain on your batteries you have to divide your nominal battery voltage (24v) into the voltage of the load (120v), which is 5, And then multiply this times your 120vac amp hours (5 x12 ah). So in this case the calculation would be 60 amp hours drained from your batteries - not the 12 ah. Another simple way is to take the total watt-hours of your 120VAC device and divide by nominal system voltage. Using the above example; 3 amps x 120 volts x 4 hours = 1,440 watt-hours divided by 24 DC volts = 60 amp hours. Lead-acid batteries are the most common in PV systems because their initial cost is lower and because they are readily available nearly everywhere in the world. There are many different sizes and designs of lead-acid batteries, but the most important designation is that they are deep cycle batteries. Lead-acid batteries are available in both wet-cell (requires maintenance) and sealed no-maintenance versions. AGM and Gel-cell deep-cycle

batteries are also popular because they are maintenance free and they last a lot longer.

Inverter: The inverter transforms the solar-produced DC electricity into the AC electricity commonly used in most homes for powering lights and appliances (Fig. 1.4). Grid-tie inverters synchronize the electricity they produce with the grid's "utility grade" AC electricity, allowing the system to feed solar-made electricity to your home and to the utility grid. Battery-based inverters for off-grid or grid-tie use often include a battery charger, which is capable of charging a battery bank from either the grid or a backup generator during cloudy weather.

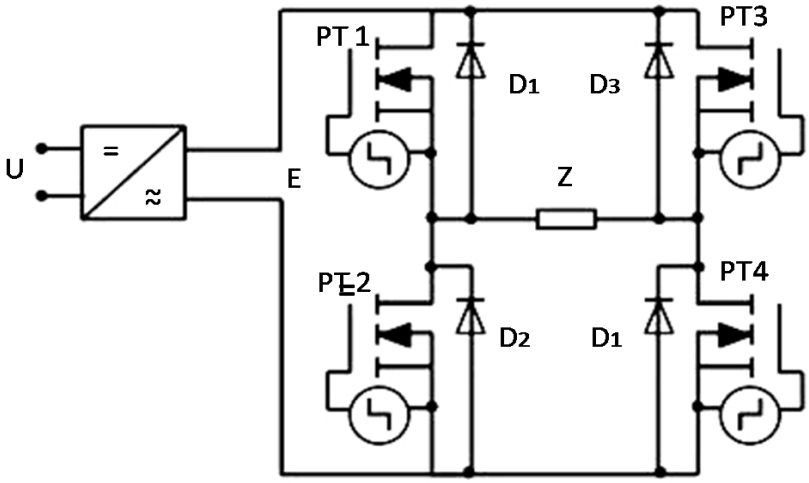

Figure 1.4. Example for inverter

The Power Inverter

Unless you plan on using battery power for everything, you will need a Power Inverter. Since the majority of modern conveniences all run on 120 volts AC, the Power Inverter will be the heart of your Solar Energy System. It not only converts the low voltage DC to the 120 volts AC that runs most appliances, but also can charge the batteries if connected to the utility grid or a AC Generator as in the case of a totally independent stand-alone solar power system.

Square Wave power inverters: This is the least expensive and least desirable type. The square wave it produces is inefficient and is hard on many types of

equipment. These inverters are usually fairly inexpensive, 500 watts or less, and use an automotive cigarette lighter plug-in. Don't even consider one of these types of power inverters for a home system.

*Modified Sine Wave power inverters:*This is probably the most popular and economical type of power inverter. It produces an AC waveform somewhere between a square wave and a pure sine wave. Modified Sine Wave inverters, sometimes called Quasi-Sine Wave inverters are not real expensive and work well in all but the most demanding applications and even most computers work well with a Modified Sine Wave inverter. However, there are exceptions. Some appliances that use motor speed controls or that use timers may not work quite right with a Modified Sine Wave inverter. And since more and more consumer products are using speed controls & timers, I would only recommend this type of inverter for smaller installations such as a camping cabin.

*True Sine Wave power inverters:*A True Sine Wave power inverter produces the closest to a pure sine wave of all power inverters and in many cases produces cleaner power than the utility company itself. It will run practically any type of AC equipment and is also the most expensive. Many True Sine Wave power inverters are computer controlled and will automatically turn on and off as AC loads ask for service. I believe they are well worth the extra cost. I use a True Sine Wave power inverter myself and find that its automatic capabilities makes it seem more like Utility Company power. The Xantrex 2500 watt power inverter I use has a search feature and checks every couple of seconds for anything that wants AC, then it powers up automatically. You just flick on a light switch (or whatever) and it works. When you turn off the light or the refrigerator kicks off for example, the power inverter shuts down to save battery power.

While the Modified Sine Wave inverter (sometimes called a Quasi Sine Wave inverter) is nearly half the price of a True Sine Wave inverter, I would still recommend using a True Sine Wave inverter if you want to supply automatic power to a normal home using a wide variety of electrical devices. Also, most

appliances run more efficiently and use less power with a True Sine Wave inverter as opposed to a Modified Sine Wave power inverter.

Grid Tie Power Inverters: If you are connected to normal Utility company power and just want to add some Free Sun Power electricity to reduce your electric bill and you do not need a totally independent system, it is possible that a Grid Tie power inverter will suit your needs. With a Grid Tie power inverter, whatever electricity that your solar panels produce will reduce the amount supplied by the utility company, in effect lowering your bill. And, if you are producing more power than you are using, you can actually sell the extra power back to the utility company! For this type of setup a much smaller battery bank can be installed just to cover short term outages from a few minutes to an hour or two. In fact, if you don't have frequent long term power outages and don't need back-up power, then you will not need any batteries at all.

Input voltages: The main consideration when deciding on the input voltage (from your battery bank) of your Inverter is the distance between your solar panel array and your battery bank. The higher the voltage, the lower the current and the smaller the (expensive) cables need to be. Of course, when you decide on a system voltage, the Solar Panels, Inverter, and Battery Bank all need to use the same voltage. More detailed information on voltage & current is explained in the tutorial on Power & Watts.

To help decide on which voltage to use, check out our Wire Size Calculator which can tell you what size wire is needed to connect the solar panels to your equipment area. You can try all 3 different voltages to see the change that it can make in wire size.

Inverter Stacking: Using multiple inverters: Two inverters can be installed in a configuration known as stacking that can provide more power or higher voltage. If two compatible inverters are stacked in series you can double the output voltage. This would be the technique to use to provide 120/240 volts AC. On the other hand, if you configure them in parallel, you can double your power. Two 4000 watt inverters in parallel would give you 8000 watts (8KW) of electricity

Power Inverter considerations: The Power Inverter is connected directly to the batteries and the main AC breaker panel to supply power from the batteries to the loads (appliances). Check out Wires & Cables for more info on the necessary wire size for installing one or use our new Wire Size Calculator. The Power Inverter converts the low voltage DC to 120 volts AC. Power Inverters are available for use on 12, 24, or 48 volt battery bank configurations. Most Power Inverters can also charge the batteries if connected to the AC line. Alternatively, the AC line input could be your own AC Generator in the case of a stand-alone solar power system. When using a AC Generator to charge the batteries, the Power Inverter transfers the AC Generator power to the loads via a relay. This way the AC Generator not only charges the batteries but also supplies your AC power while it is running. If your Generator is at least 5000 watts, you can charge your batteries and have extra AC power at the same time.

An inverter is a device which changes DC power stored in a battery to standard 120/240 VAC electricity (also referred to as 110/220). Most solar power systems generate DC current which is stored in batteries. Nearly all lighting, appliances, motors, etc., are designed to use ac power, so it takes an inverter to make the switch from battery-stored DC to standard power (120 VAC, 60 Hz). In an inverter, direct current (DC) is switched back and forth to produce alternating current (AC). Then it is transformed, filtered, Stepped, etc. to get it to an accept able output wave form. The more processing, the cleaner and quieter the output, but the lower the efficiency of the conversion. The goal becomes to produce a waveform that is acceptable to all loads without sacrificing too much power into the conversion process [11].

Inverters come in two basic output designs

- Sine wave and modified sine wave. Most120VAC devices can use the modified sine wave, but there are some notable exceptions. Devices such as laser printers which use triacs and / or silicon controlled rectifiers are damaged when provided mod-sine wave power. Motors and power supplies usually run warmer and less efficiently on mod-sine wave power. Somethings, like fans, amplifiers,

and cheap fluorescent lights, give off an audible buzz on modified sine wave power. However, modified sine wave inverters make the conversion from DC to AC very efficiently.They are relatively inexpensive, and many of the electrical devices we use every day work fine on them. Pure sine wave inverters can virtually operate anything. Your utility company provides sine wave power, so a sine wave inverter is equal to or even better than utility supplied power. A sine wave inverter can "clean up" utility or generator supplied power because of its internal processing. Inverters are made with various internal features and many permit external equipment interface. Common internal features are internal battery chargers which can rapidly charge batteries when an AC source such as a generator or utility power is connected to the inverter's INPUT terminals. Auto-transfer switching is also a common internal feature which enables switching from either one AC source to another and / or from utility power to inverter power for designated loads. Battery temperature compensation, internal relays to control loads, automatic remote generator starting / stopping and many other programmable features are available.

Most inverters produce 120VAC, but can be equipped with a step-up transformer to produce 120 / 240VAC. Some inverters can be series or parallel "stacked-interfaced" to produce 120 / 240VAC or to increase the available amperage.

Efficiency Losses: In all systems there are losses due to such things as voltage losses as the electricity is carried across the wires, batteries and inverters not being 100 percent efficient, and other factors. These efficiency losses vary from component to component, and from system to system and can be as high as 25 percent.

1.4. Types of autonomous solar power system

The modularity and flexibility of solar electricity allows users to have a system tailored to their specific needs and preferences. In addition to full or partial

power for a home or business, solar electricity may serve as a power source for a specific job. This could be electricity for a well pump, patio or street lighting, or for a home security system or even a backyard waterfall. Typically, such systems consist of one or more modules and a charge controller accompanied by a battery or batteries.

Generally speaking, solar power systems may be categorized into three primary types: standalone, battery backup, and utility (grid) connected. Any of these types of systems may be designed to meet all or part of the user's electrical requirements.

Photovoltaic (PV) systems are composed of several individual components including arrays (multiple connected modules), inverters, controls, safety disconnects, and batteries. By assembling differing sizes of components together, systems can be built with varied power outputs to meet the demands of various loads.

Some applications need a system that includes a fuel power backup generator, wind turbine or water turbine. Typically, such "hybrid" systems share the load between the solar chargers and the back-up generator. Batteries are still required, plus the DC to AC inverter if regular AC loads will be powered.

Autonomous DC System

Autonomous type systems are usually a utility power substitute. They generally include solar charging modules, storage batteries and controls including a charge regulator. A small standalone DC system is an excellent replacement for kerosene lamps and noisy generators in a remote home, a recreational vehicle, or a boat. The actual sizing depends on the wattage of the loads and how often they are to be run.

In this system a photovoltaic (PV) array charges the battery during daylight hours and the battery supplies power to the loads when needed. The charge regulator terminates the charging when the battery reaches full charge.

Autonomous AC-DC System

This system is the same as the previous system, except for the use of a DC to AC inverter. With the addition of an inverter, commonly available household appliances such as computers, power tools, vacuum cleaners, washing machines and kitchen appliances can be solar powered.

High quality DC to AC inverters are available with power outputs ranging from one hundred watts to ten kilowatts and more, and conversion efficiencies greater than 90 percent. Most larger inverters also have the ability to serve as battery chargers from a backup generator when more power is needed than can be supplied by the solar modules.

Backup AC System

A backup AC solar electric system will usually have a photovoltaic (PV) array of ten or more modules, a battery bank, and one or more inverters. The utility will backup the solar and run the loads when available and needed. If utility power fails, the power from the battery bank is available to the system.

Utility Interconnected System (Grid-tied)

These are the simplest systems and require no batteries because they are designed not for back-up power but to contribute power back into the existing power supply.

By lowering a building's power bills, these systems will pay for themselves over a number of years and reduce the air pollution produced by utility companies that burn coal. Contributing clean, green power from your own roof helps create jobs and in sunny states like Florida is the best alternative to buying electricity derived from fossil fuels.

Utility interconnected systems are generally designed to reduce power demands from the utility by 'net metering' power or in some cases to sell power back to the utility. A typical system might include solar modules, a mounting structure, and AC inverter/control for the power to be fed back through the building's 120/208/240 volt AC power distribution system.

In grid-connected or grid-tied systems, solar energy is used during the day by the system owner. At night, the owner draws on the previously established

electricity grid. An addition benefit of the grid-tied system is that the solar system does not need to be sized to meet peak loads—overages can be drawn from the grid. In many cases, surplus energy generated during the day can be exported back to the grid. Grid-connected systems must meet utility requirements. For example, inverters must not emit noise that can interfere with equipment reception. Inverters must also switch off in cases of grid failure. Finally, they must retain acceptable levels of harmonic distortion, such as voltage quality and current output waveforms.

Grid-connected systems can be applied to residential installations.

AutonomousGrid-Tied Systems

Stand-alone grid-connected systems are the same as grid-connected systems, except with battery storage added to allow power to be generated even if the electricity grid fails.

Stand-alone grid-tied systems can be applied to residential and business systems that require uninterrupted power.

Off-Grid Systems

Off-grid systems are not connected to the electricity grid. The output of an off-grid system is entirely dependent upon the intensity of the sun. The more intense the sun exposure, the greater the output. The electricity generated is used immediately, so the system must function on direct current and variable power output.

Off-grid systems can be used for water pumps and greenhouse ventilation systems. Specialized solar water pumps are designed for submersible use (in a borehole) or to float on open water

Autonomous Off-Grid Systems

If a certain power output guarantee is required at any time of the day or night, either some kind of storage device is necessary, or the PV system should be combined with another energy supply such as propane or a diesel generator (see hybrid systems, below). Most off-grid systems use batteries to store power during periods of low to no sunlight.

Stand-alone off-grid systems can be applied to remote homes, lighting, TV, radio, and telemetry.

Autonomous Off-Grid Hybrid Systems

To meet the largest power requirements in an off-grid location, the PV system can be configured with a small diesel generator. This means that the PV system no longer has to be sized to cope with the worst sunlight conditions available during the year. Use of the diesel generator for back-up power is minimized during the sunniest part of the year to reduce fuel and maintenance costs.

Conclusion to chapter

In this unit we have seen how had organized the solar power system. Also I read about components and how they work at the station. And how connect them to each other. While writing this unit I got to know with types of solar power stations. After this we will know advantages and benefits of solar power system. It is very good way to get energy for free. That's way I decided to use this system on the telecommunication objects. I mean they will work automatically. This project is very enduring. Because it has very quality guaranty. This project will work 25 years without any energy from outside.

2. MEASURING PARAMETER OF AUTONOMOUS SOLAR POWER SYSTEM

2.1. Parameter of humidity

If we talk about the energy which is received from Sun, Earth receives approximately of 1413 W/m2 and the actual consumption which appears on the scale formulated is approximately 1050W/m2 as recorded by Pacific Northwest Forest and Range Experiment Station Forest Service, U.S. Department of Agriculture, Portland, Oregon, USA in in 1972.As per the facts observed, approximately of 30% energy is lost in between. As per the statistical figures stated, that the Earth's top of the atmosphere sunlight's intensity is about 30% more intense than the actual received on the land. In the Solar panels what we use today, actually we make use of the 70% energy coming from the Sun and utilize the working of our panels to fulfill our energy needs [9].

As per the fact that the earth's crust mainly consists of 70% of Water, the energy which strikes the earth is indirectly striking the water/oceans which helps in increasing of humidity level on the overall basis. The humidity doesn't only create hurdles for the energy actually received at the top of the atmosphere but also effects the device consumptions by many aspects.

The aspects what we covered is the effect of humidity on the Solar panels which create obstacles for drastic variation in the power generated, indirectly making the device work less efficient than it could have without it. The cities where in the humidity level is above the average range of 30 actually results in the minimal layer of water on the top of the Solar panel which results in decreasing of the efficiency.

As per the facts when the light consisting of energy/Photon strikes the water layer which in fact is denser, Refraction appears which results in decreasing of intensity of the light which in fact appears the root cause of decreasing of efficiency.

Additional there appears minimum components of Reflection which also appears on the site and in that, there appears light striking is subjected to more losses which after the experiments conducted resulted approximately in 30% loss of the total energy which is not subjected to utilization of Energy for the Solar panel.

AS far as the efficiency of the Solar cell is concerned, Efficiency is termed as the amount of the light that can be converted into usable format of electricity. Because of the efficiency depends upon the value of Maximum Power Point of the Solar cell , due to the above effect of humidity ,the maximum power point is deviated and that indirectly results in decreasing of the Solar cell Efficiency.

The usage of the Solar Panel is readily effected by the effect of humidity and the values corresponds to change is the humidity is subjected to change Experiment and Analysis:

Various experiments were conducted and in the test bench included 50W BP Solar Panel having specification of Vamp = 17.3V and Imp = 2.9A with temperature coefficient of Isc= (0.065 ± 0.015) % / °C, Temperature coefficient of Voc=-(80 ± 10) mv / °C and Temperature coefficient of power=-(0.5 ± 0.05) % / °C, and Tungsten Halogen Bulb of 1000W, 2 Humidifier, Hygrometer, Thermometer, Output Load as tungsten filament bulbs(15,20,25W), 2 Millimeters [8].

Results were calculated initially with normal temperature in Karachi which was 32*C (305K) and humidity 25.The humidifier was used as to increase the humidity level of the area where in the Solar panel was connected with the load and was subjected to a constant intensity by using tungsten Halogen bulb and the distance was kept 2 foot.

The results showed the drastic change in the readings when the humidity was subject to gradually increase [Table 2.1]. Below is the chart in where the readings are note [5].

Table 2.1

Humidity vs. voltage, current and power readings taken through the experimental set up as discussed

Temperature (K)	Humidity(%)	Voltage (DC)	Current Amps(DC)	Powers(watts)
32	25	17.10	2.78	47.538
32	30	16.72	2.63	43.973
32	35	16.53	2.42	40.002
32	40	16.45	2.3	37.605
32	45	16.41	2.14	35.117
32	50	16.33	2.04	33.313
32	55	16.32	1.88	30.681

2.2. Temperature

When the temperature dependence of solar cell characteristics is concerned, although an increase of the current with the temperature increase was observed, main output characteristics such as efficiency were negatively influenced by high temperature. This is due to the fact that open circuit voltage rapidly decreases with an increase of the temperature, as could be seen in Fig. 2.1.

The rate of the decrease for this particular cell was -2,48mV/0 C (using linear approximation method), and that made it particularly temperature sensitive. Also, because of the working conditions of solar cells (direct exposure to the solar radiation), their temperature increases very rapidly (up to 40 C in the first 2-3 minutes of work), so better temperature stabilization of characteristics, and/or adequate cooling are main requirements for successful and long-term operation of solar systems. This is specially important because for non-ideal devices ideality factor n is greater than 1, indicating more complex temperature dependence of basic properties such as diffusion length or charge carrier lifetime [10].

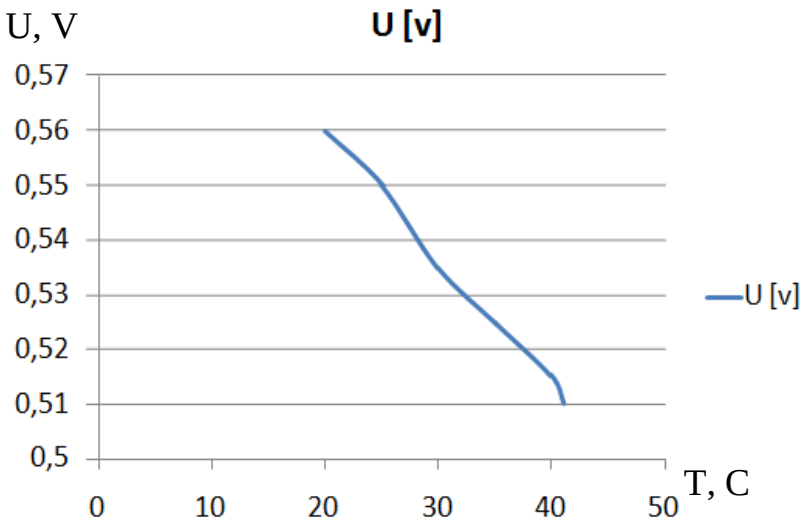

Figure 2.1. Example for temperature dependence of open circuit voltage

One of the major performance limitations of photodiodes is the degradation of electrical and optical characteristics of the photodetectors in the increased temperature conditions. First part of the paper was oriented to the frequency dependent 1/f noise in contacts, since temperature increase induces higher level of noise. It was established that both physical and electrical properties of used silicides are influenced by the implantation doses. But the results of frequency noise measurements indicate that ion implantation could successfully be applied in order to achieve a more homogeneous silicidation and very good temperature stability, if carefully optimized dose was used.

2.3. Measuring voltage and current

Voltage (energy per charge) is related to the energy delivered by each charge. Two PV cells connected in series increases the energy for each charge. However, the current remains constant. The power delivered is higher since the energy per charge is greater and but the number of changes remains the same. This is one reason why solar cells are connected in series in Solar Panels, to be able to provide enough voltage to meet the rated requirements.

The open-circuit voltage, V_{OC}, is the maximum voltage available from a

solar cell, and this occurs at zero current. The open-circuit voltage corresponds to the amount of forward bias on the solar cell due to the bias of the solar cell junction with the light-generated current. The open-circuit voltage is shown on the IV curve below.

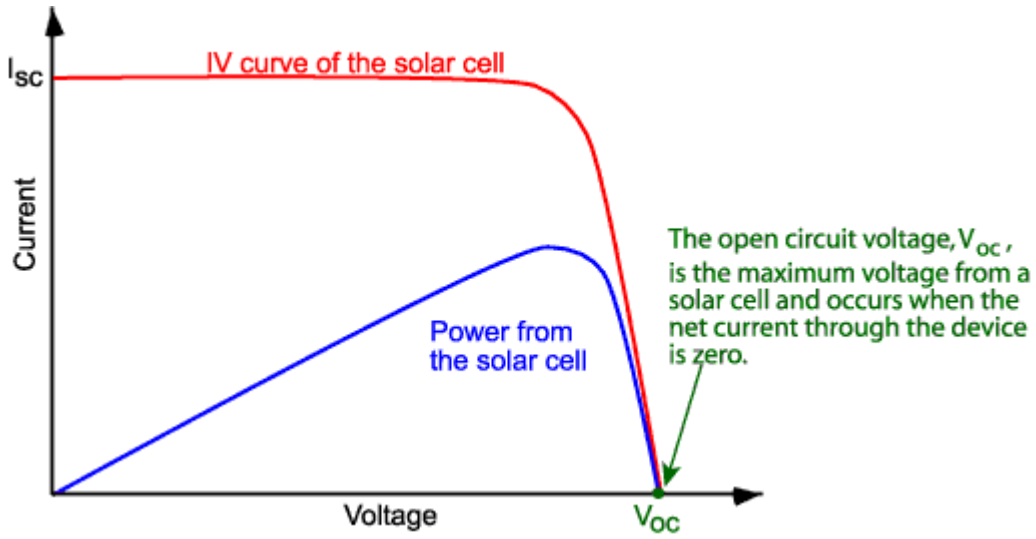

Figure 2.2. IV curve of a solar cell showing the open-circuit voltage

An equation for V_{oc} is found by setting the net current equal to zero in the solar cell equation to give:

$$V_{oc} = \frac{nkT}{q} \ln\left(\frac{I_L}{I_0} + 1\right). \tag{2.1}$$

The above equation shows that V_{oc} depends on the saturation current of the solar cell and the light-generated current. While I_{sc} typically has a small variation, the key effect is the saturation current, since this may vary by orders of magnitude. The saturation current, I_0 depends on recombination in the solar cell. Open-circuit voltage is then a measure of the amount of recombination in the device. Silicon solar cells on high quality single crystalline material have open-circuit voltages of up to 730 mV under one sun and AM1.5 conditions, while commercial devices on multicrystalline silicon typically have open-circuit voltages around 600 mV.

The V_{OC} can also be determined from the carrier concentration:

$$V_{oc} = \frac{kT}{q} \ln\left[\frac{(N_A + \Delta n)\Delta n}{n_i^2}\right] , \qquad (2.2)$$

where kT/q is the thermal voltage, N_A is the doping concentration, Δn is the excess carrier concentration and n_i is the intrinsic carrier concentration. The determination of V_{OC} from the carrier concentration is also termed Implied V_{OC}.

At a given level of irradiance, the amount of power generated by the solar cell varies dependent upon the load. By varying the load from zero at short circuit current to infinity at the open circuit limit all possible combinations of current and voltage are traced out [Fig. 2.3]. This is called the IV curve.

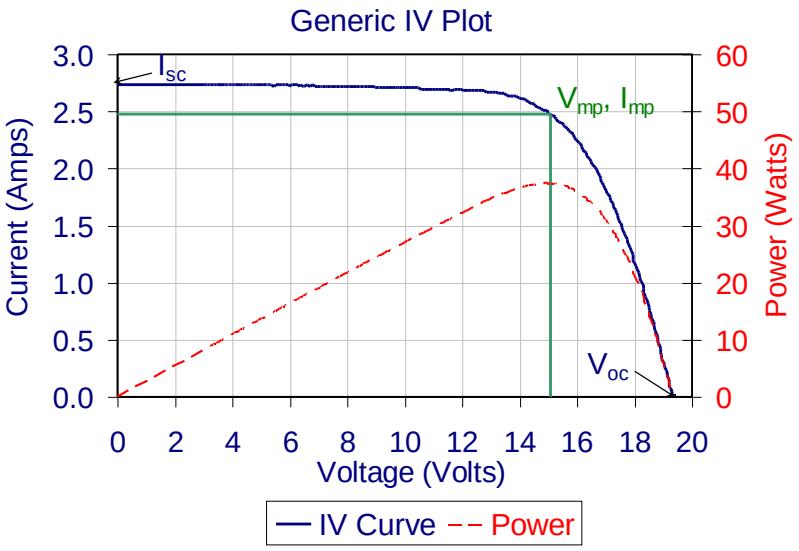

Figure 2.3. Plot of current verses voltage for a solar cell module

Power is the product of current times voltage. The power scale is on the right. The voltage and current where their product produce the most power is called the max power point and V_{mp} and I_{mp} are the max power voltage and current respectively. Short circuit current is labeled I_{SC} and open circuit voltage is labeled V_{OC}.

The load on the circuit determines the voltage and current. Depending on the position on the IV curve, an increase in load can increase or decrease the power output by the solar cells. Doubling the number of cells will generally increase the power, but the power increase will depend on the current and voltage dictated by the load. Modern inverters contain max power point trackers that adjust the perceived load to operate the facility at maximum possible output. A photovoltaic system connected to the utility grid contains an inverter that changes direct current and voltage from the solar array to alternating current that is available on the utility grid.

When two cells are connected in parallel, the current doubles and the voltage stays the same. When the two cells are connected in series, the voltage doubles and the current stays the same. Either way, the power doubles.

The response of the load to doubling the current or voltage is dependent on the type and characteristics of the load.

2.4. Measurement of solar radiation

In PV system design it is essential to know the amount of sunlight available at a particular location at a given time. The two common methods which characterise solar radiation are the solar radiance (or radiation) and solar insolation. The solar radiance is an instantaneous power density in units of kW/m^2. The solar radiance varies throughout the day from 0 kW/m^2 at night to a maximum of about 1 kW/m^2 [14]. The solar radiance is strongly dependant on location and local weather. Solar radiance measurements consist of global and/or direct radiation measurements taken periodically throughout the day. The measurements are taken using either a pyranometer (measuring global radiation) and/or a pyrheliometer (measuring direct radiation). In well established locations, this data has been collected for more than twenty years.

An alternative method of measuring solar radiation, which is less accurate but also less expensive, is using a sunshine recorder [Fig. 2.4]. These sunshine

recorders (also known as Campbell-Stokes recorders), measure the number of hours in the day during which the sunshine is above a certain level (typically 200 mW/cm^2). Data collected in this way can be used to determine the solar insolation by comparing the measured number of sunshine hours to those based on calculations and including several correction factors.

Figure 2.4. Equipment for solar irradiance measurements

A final method to estimate solar insolation is cloud cover data taken from existing satellite images.

While solar irradiance is most commonly measured, a more common form of radiation data used in system design is the solar insolation. The solar insolation is the total amount of solar energy received at a particular location during a specified time period, often in units of $kWh/(m^2\,day)$. While the units of solar insolation and solar irradiance are both a power density (for solar insolation the "hours" in the numerator are a time measurement as is the "day" in the denominator), solar insolation is quite different than the solar irradiance as the solar insolation is the instantaneous solar irradiance averaged over a given time

period. Solar insolation data is commonly used for simple PV system design while solar radiance is used in more complicated PV system performance which calculates the system performance at each point in the day. Solar insolation can also be expressed in units of MJ/m^2 per year [15].

Solar radiation for a particular location can be given in several ways including:

– typical mean year data for a particular location
– average daily, monthly or yearly solar insolation for a given location
– global isoflux contours either for a full year, a quarter year or a particular month
– sunshine hours data
– solar Insolation Based on Satellite Cloud-Cover Data
– calculations of Solar Radiation

Conclusion to chapter

On the other hand, from the I-V measurements obtained data have shown that though there is significant increase of solar cells current with an increase of temperature, other electrical characteristics rapidly degrade leading to the decrease of the efficiency. Owing to the strong correlation between burst noise and excess current, degradation of both electrical and optical output characteristics of the device could be monitored through the ideality factor. Observed increase of the ideality factor with the temperature, indicates an increase of the current noise and detection threshold, and decrease of the resolution of the photodetector device. For this reason monitoring of the device characteristics should be performed continuously, especially because solar cells are exposed to the severe working conditions such as increased temperature.

3. MODEL OF THE MONITORING SYSTEM FOR AUTONOMOUS SOLAR POWER PLANTS

3.1. Sensors and components for autonomous solar power system

The main feature of this mechanism is continues monitoring of solar system as well as controlling action will take place against faults occurring in some parameter.

Of, course we will put some sensors detector to know any changes at solar power panels.

Temperature Sensor

The TMP36 are low voltage, precision centigrade temperature sensors [Fig.3.1]. They provide a voltage output that is linearly proportional to the Celsius (centigrade) temperature.

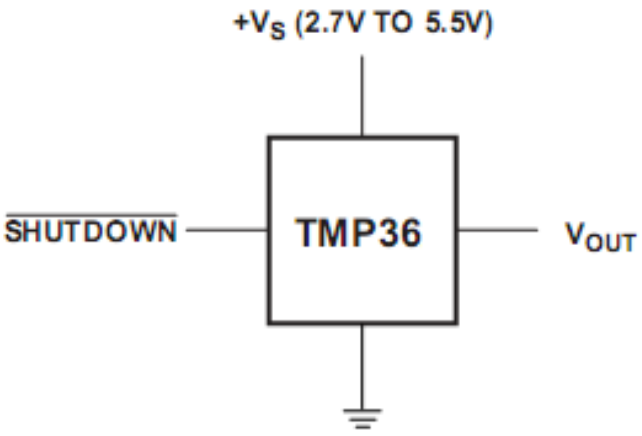

Figure 3.1. TMP 36 temperature sensor

The TMP36 do not require any external calibration to provide typical accuracies of ±1°C at +25°C and ±2°C over the −40°C to +125°C temperature range. The low output impedance of the TMP36 and its linear output and precise calibration simplify interfacing to temperature control circuitry and ADCs. All three devices are intended for single-supply operation from 2.7 V to 5.5 V

maximum. The supply current runs well below 50 μA, providing very low self-heating—less than 0.1°C in still air. In addition, a shutdown function is provided to cut the supply current to less than 0.5 μA.

The TMP36 is specified from −40°C to +125°C, provides a 750 mV output at 25°C, and operates to 125°C from a single 2.7 V supply. The TMP36 is functionally compatible with the LM50. Both the TMP35 and TMP36 have an output scale factor of 10 mV/°C.

Irradiation Sensor circuit, such as the name is a circuit that utilizes the light intensity / beam for the other applications do automatically via the relay to switch on / off under certain conditions. Applications such as these and more we encounter in everyday life.

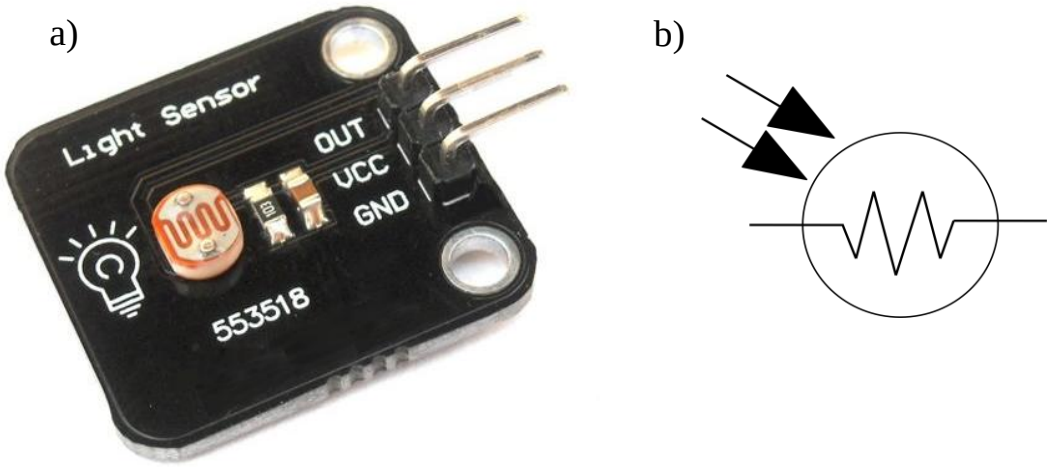

Figure 3.2. a) Light sensor; b) Light Dependent Photodiode

A Light sensor requires a light detector which can be a Light Dependent Resistor (LDR) or Photodiode [Fig. 3.2]. The circuit measures the ambient light intensity is called ambient light sensor circuit. Basically, it can simply be a light sensor circuit consists of three parts, namely the ambient light sensor, relay driver, and application design. So it can be concluded that a light sensor circuit is a light sensor switch circuit.

Normally the resistance of an LDR is very high, sometimes as high as 1000000 ohms, but when they are illuminated with light resistance drops dramatically. It also is capable of reacting to a broad range of frequencies (Infrared (IR), visible light and ultraviolet (UV)).

Humidity Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output [Fig.3.3]. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost effectiveness.

Figure 3.3. DHT11 humidity sensor

Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The component is 4-pin single row pin package. It is

convenient to connect and special packages can be provided according to users' request. In the following table shown detailed specifications [Table 3.1].

Table 3.1

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 bit	
Accuracy	25 ^{o} C		±4%RH	
	0 ^{o}-50 ^{o} C			±5%RH
Measurement Range	0 ^{o}C	30%RH		90%RH
	25 ^{o}C	20%RH		90%RH
	50 ^{o}C	20%RH		80%RH
Response Time (sec)	1/e(63%)25 ^{o}C 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	

Relative Humidity (RH) measured by partial pressure divide to vapor pressure or

$$RH= PP/VP *100\% . \quad (3.1)$$

Current Sensor is a device that detects electric current in a wire, and generates a signal proportional to it. The generated signal could be analog voltage or current or even digital output. It can be then utilized to display the measured current in an ammeter or can be stored for further analysis in a data acquisition system or can be utilized for control purpose.

The following sensor measure up to 30 amps DC or ±30 amps AC with this current sensor [Fig. 3.4]. Separate outputs for AC and DC components connect to analog input.

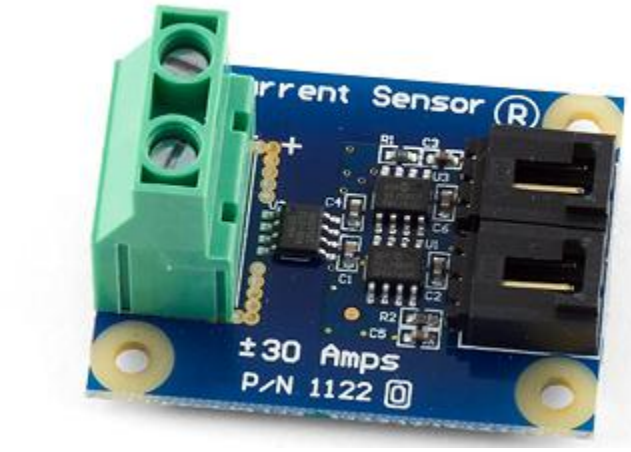

Figure 3.4. Current sensor

The 1122 measures AC current up to 30 Amps and DC Current from -30Amps to +30Amps. Dual outputs allow the user to measure both the AC and DC components of complex current waveforms separately.

Voltage sensor measure AC and/or DC voltage levels. They receive voltage inputs and provide outputs as digital voltage signals. The following module is based on resistance points pressure principle, and it can make the input voltage of red terminal reduce 5 times of original voltage [Fig. 3.5].

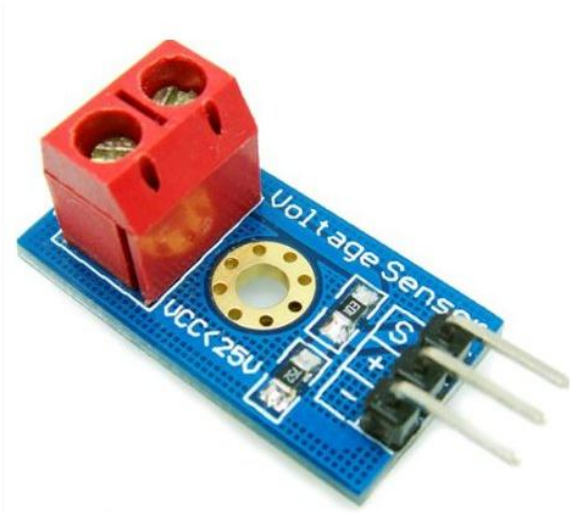

Figure 3.5. Voltage sensor

The max Arduino analog input voltage is 5V, so the input voltage of this module should be not more than 5V x 5= 25 V (if for 3.3 V system, the input voltage should be not more than 3.3 V x 5 = 16.5 V).

Because the Arduino AVR chip have 10 bit AD, so this module simulation resolution is 0.00489 V (5 V / 1023), and the input voltage of this module should be more than 0.00489 V x 5 = 0.02445 V.

Solar Irradiance Measurements

Irradiance refers to the power of solar radiation per unit area on a surface. Global irradiance on a horizontal surface is comprised of direct irradiance and diffuse irradiance. On a tilted surface, the reflected irradiance from the ground also contributes to the total global irradiance. Generally, solar irradiance is called insolation.

Solar radiation affects many systems in the house and can vary considerably within the same town. On-site, solar irradiance is a particularly useful measurement if there is a photovoltaic (PV) system or solar thermal system installed at the field test location. Also, if space conditioning is a focus of the field test, solar irradiance is an important measurement as solar radiation has a large effect on heating and cooling load requirements, electrical lighting demand (day lighting), and envelope performance.

There are three types of sensors can be used to measure irradiance, all with a specific purpose: Pyranometers, Pyrgeometers and Photometers.

A pyranometer is a device used to measure solar irradiance, or insolation, both direct and diffuse on a planar surface [Fig 3.6]. Pyranometers are typically passive devices, requiring no power to operate. Pyranometers use a black-coated thermopile that will absorb all solar radiation across a wide wavelength range. A glass dome limits the radiation to the short wave range only. The thermopile generates a voltage signal that is proportional to the incident solar radiation. Alternatively, a silicon photovoltaic detector can be used to measure the incoming radiation for wavelengths, which generates a current signal that is converted to a

voltage signal using a potentiometer. The variable resistance of the potentiometer is used to calibrate the sensitivity for the photovoltaic sensors.

a) Thermopile-type pyranometer

b) Photovoltaic detector-type pyranometer.

Figure 3.6. Pyranometers

Pyranometers are a common sensor for field tests, especially when photovoltaic (PV) panels are installed at the house, or cooling loads are a main subject of the field test. Typically, two pyranometers will be installed: one mounted on a horizontal surface and a second affixed to one of the PV panels (if applicable) in order to measure the available short wave radiation in the plane of the collector. The horizontally mounted pyranometer should be leveled using adjustable feet and the level sensor (shown in the photo above, right). Pyranometers can be used to determine the installed efficiency of PV systems, since the available solar radiation in the plane of the PV panels will be measured. Additionally, insolation can have a large effect on the cooling and heating loads in the house and capturing the on-site insolation, in addition to outdoor temperature and humidity, will help complete the picture of the environmental effects on the home's HVAC system.

Arduino

The Arduino Uno is a microcontroller board based on the ATmega328 [Fig. 3.7]. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an

ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to serial converter.

Figure 3.7. Arduino Uno

1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes. In the following shown features [Table 3.2].

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow

you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

Table 3.2

Features of ArduinoUno

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)

3.2. Model of solar power station

Solar power systems allow you to generate all your own power and do not require a connection to the power grid [Fig. 3.8]. These systems are popular with customers that want to be self sufficient and or, are not close to the power grid. The solar power is generated by solar panels that can be fixed to the building's roof or on purpose built poles and stands. The power generated is used by the appliances in the building and the excess power that is generated is fed into batteries to keep them charged. At night the batteries supply power to the building which means no more power bills.

This system suits medium to large installation budgets and is widely used for remote residential properties, farms, community use buildings and mining applications that do not have access to the power grid. These systems can often incorporate a small wind turbine as well to supplement the solar power production and assist with charging the batteries.

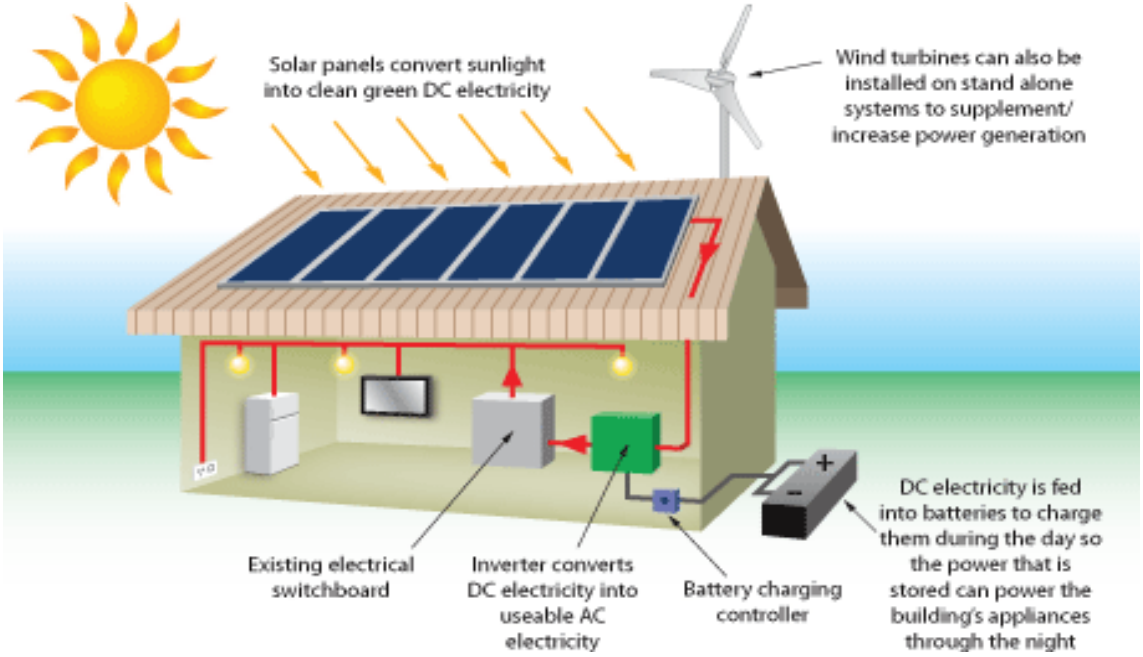

Figure 3.8. Stand Alone solar power systems

By installing a standalone solar system you can look forward to having no electricity bills and you can save more than 3 tonnes of greenhouse gas emissions from being released into the earth's atmosphere annually which is the equivalent of taking approximately 6 cars off the road for a year.

Standalone solar power systems are designed specifically to suit your power requirements. At GES we use quality materials and equipment with 25 year manufacturer's output warranties on all our panels. Our experienced team of installers are fully insured, licensed and accredited with the Clean Energy Council of Uzbekistan. We also hold accredited professional status with the Green Building Council of Uzbekistan

With the abundance of sun in Australia it makes sense to go solar. The

positive benefits are many which makes installing a stand alone solar system a sound investment.

Building a Solar System

The first and most obvious component in your solar setup is the array of panels. An important point with solar panels is that the voltage is always higher than 12V (usually around 17.5V). When you work out your panel requirement you must use the amperage rating, not wattage, as the higher voltage means lower amperage and therefore the wattage can be misleading. You should always check the specifications when buying solar panels. Example: A 100W, '12V' panel is actually 17,5V. The amperage this panel will give out is not 100W/12V, but rather 100W/17.5V = 5.7 amps.

Your next component is the charge controller, also known as the regulator. Solar regulators are different to ones used for wind power. There are hundreds of different types of regulators with sizes ranging from 5 Amps to 80 Amps. You also get different types; the normal, straight-forward regulator which is great for smaller loads, and the MPPT regulator which adds about 20% efficiency to the system and is used for bigger systems. The regulator size requirement is determined by the amount of energy supplied by the panels. The rating of the regulator is how much energy it can handle per hour. Example: A 60 amp controller will handle 60 amps per hour. Any energy in excess of that rating (in this case 60 amps) will be wasted, therefore it is important that you don't undersize your regulator.

Next, you need batteries; there are many different makes and types available. Normal marine deep cycle batteries are more commonly used for smaller systems, while the special high amperage solar batteries are ideal for larger installations. Normal car batteries will not perform well with solar power systems because of their shorter life spans. It is a bad idea to draw all the energy out of the batteries. We give them 50% depth of depletion, to prevent damage and lengthen their life span. Example: With a battery of 105 amp hours, we work on 50 amp hours of useable storage.

An inverter is the final step in your basic solar system (before the distribution board). Solar setups are always DC (12V, 24V, 36V or 48V). What the inverter does is it converts the DC voltage into AC voltage to run your normal everyday appliances. You get appliances with 12V configurations nowadays, which allow for a direct connection from the batteries to the appliances or db board, cutting out the need for an inverter. The inverter size is determined by the peak amount of energy your load can draw at any one time. For safety, you should add 20% to the inverter requirement. Example: If you're powering 5 lights of 20W each, your inverter needs to be 5 x 20W (+20%) = 120W.

Cabling, lugs, switches and connection boxes are smaller, yet essential components. The quantities are determined by the size of your system, the number of connections and the distances between components. With a 12V system, it is advisable to keep the panels and batteries within 10m of each other. The regulator and inverter should both be placed near the battery bank. The cable thickness is determined by the size of your system and the distance that the energy will travel. If you're using a 12V, 1000W system with a distance of 10m, you need 10-12mm cabling.

It is also a good idea to get a combiner box, which allows you to plug all of the wires from your panels into one box, with a single wire coming out from the other end. This just neatens the system and makes it more manageable for installation, maintenance and upgrades.

3.3. Algorithm of the autonomous monitoring center

The algorithm implementation of this system can be divided in to two main parts. PC (transmitter) side and microcontroller side (Arduino). The software used for the Arduiono is C based on Assembler software [Fig. 3.9, Fig. 3.10]. The whole programming at transmitter side is written in Java language [Fig. 3.11-3.13]. At receiver side, software implementation is based on Java and the received data is saved in SQL database.

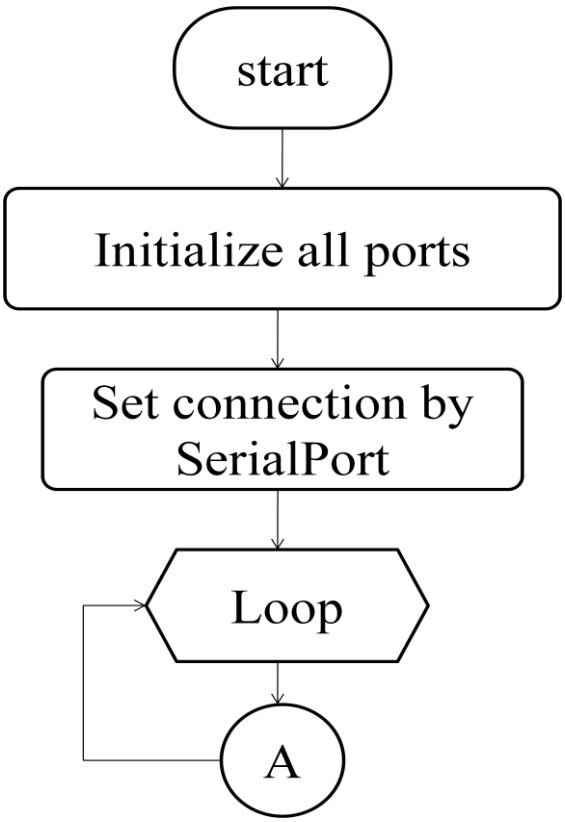

Figure 3.9. Main method for controlling sensors

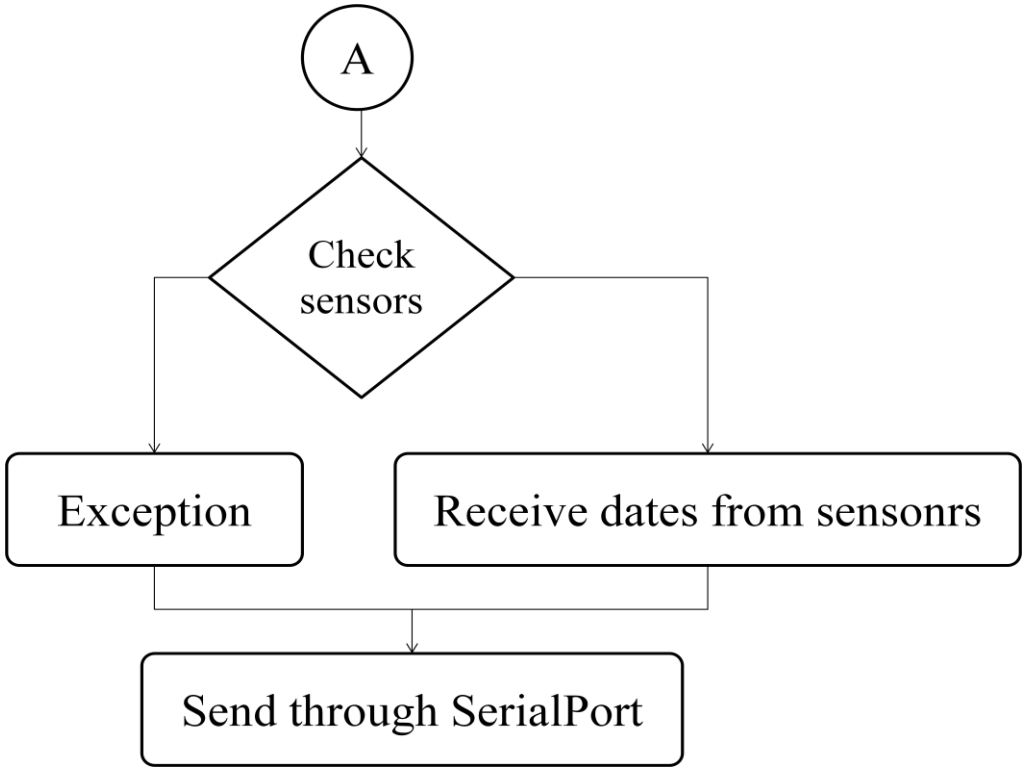

Figure 3.10. Sub method

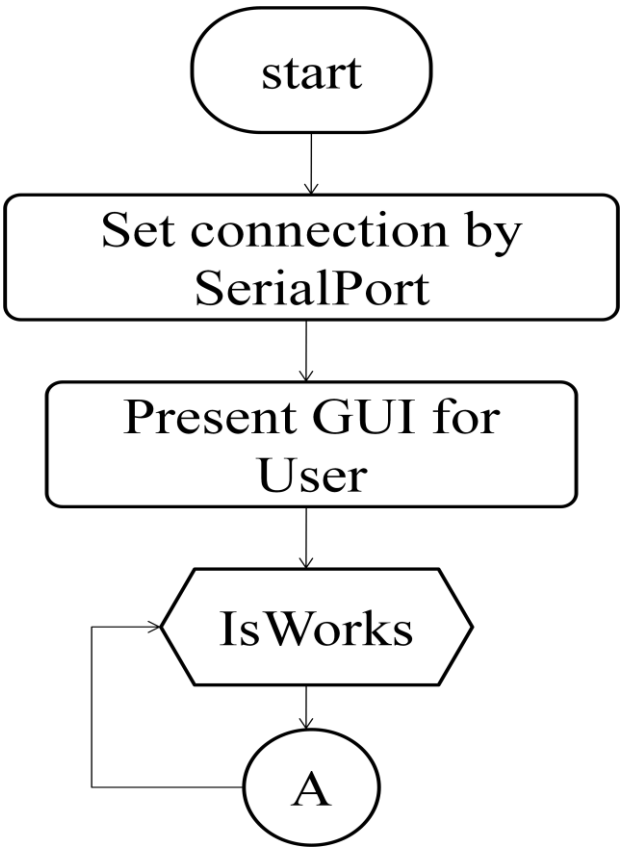

Figure 3.11. Main method for PC

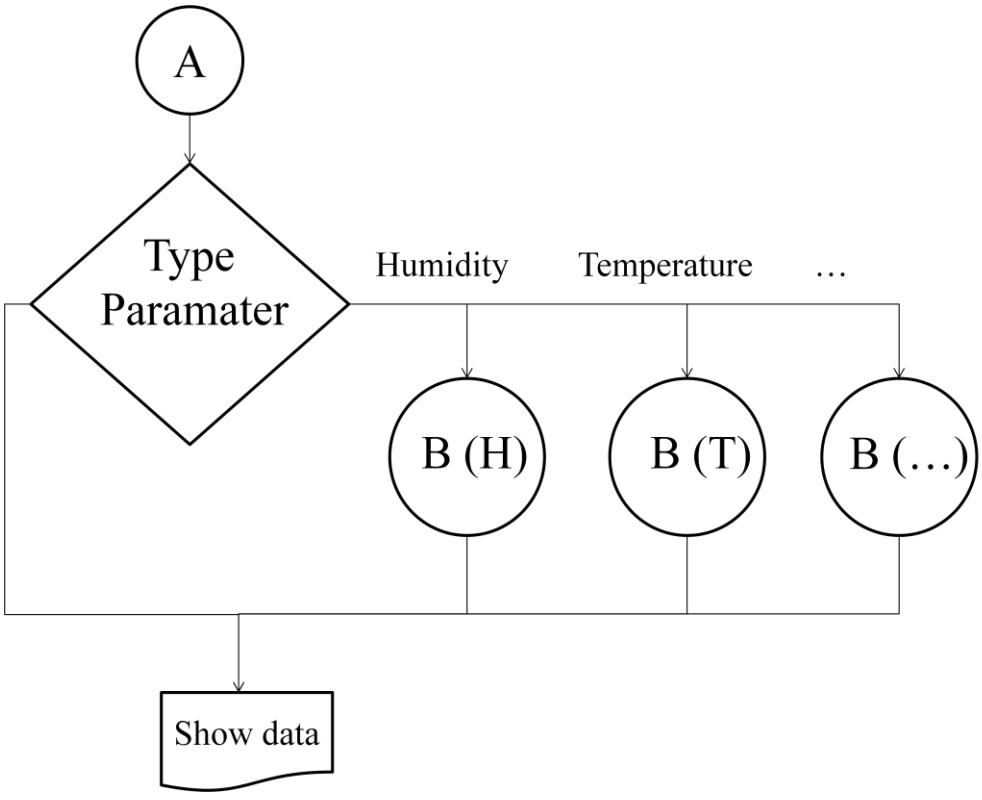

Figure 3.12. Sub method choosing parameters

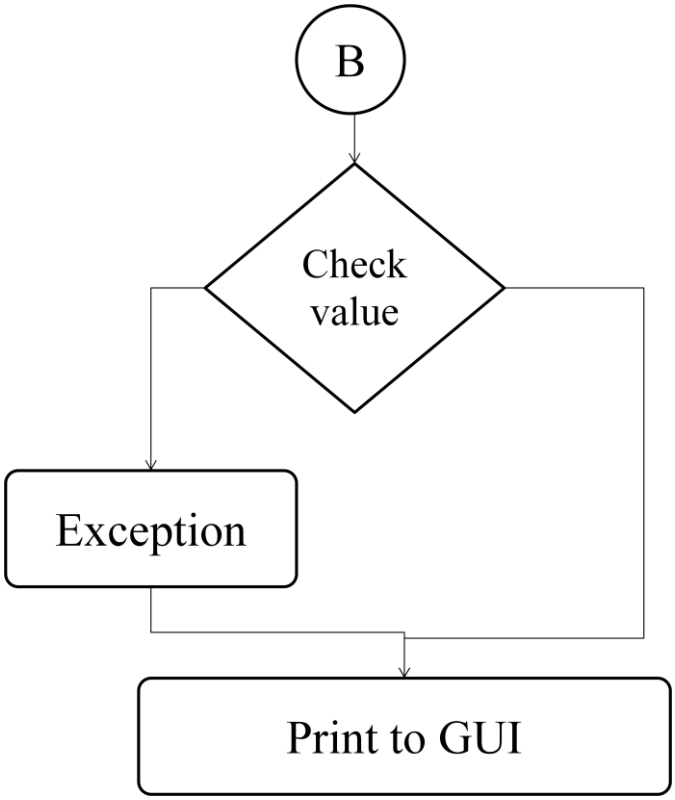

Figure 3.13. Sub method for check and show values

3.4. Developing measuring - controlling program of autonomous solar power station

A program of the local monitoring center should be work fast and correctly. And it demand more speed from local server. For making a program for local or main server firstly we have to choose useful programming language. I choose Java programming language because Java is the most popular and reached with useful classes. For example Networking I/O (for exchange information between local server to another monitoring point), SerialPort I/O (for exchanging information between maquette and local server) classes. So, what is java exactly.

Java is related to C++, which is a direct descendant of C. Much of the character of Java is inherited from these two languages. From C, Java derives its syntax. Many of Java's object-oriented features were influenced by C++. In fact, several of Java's defining characteristics come from—or are responses to—its

predecessors. Moreover, the creation of Java was deeply rooted in the process of refinement and adaptation that has been occurring in computer programming languages for the past several decades. For these reasons, this section reviews the sequence of events and forces that led to Java. As you will see, each innovation in language design was driven by the need to solve a fundamental problem that the preceding languages could not solve. Java is no exception.

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991. It took 18 months to develop the first working version. This language was initially called "Oak," but was renamed "Java" in 1995. Between the initial implementation of Oak in the fall of 1992 and the public announcement of Java in the spring of 1995, many more people contributed to the design and evolution of the language. Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin, and Tim Lindholm were key contributors to the maturing of the original prototype.

Somewhat surprisingly, the original impetus for Java was not the Internet! Instead, the primary motivation was the need for a platform-independent (that is, architecture-neutral) language that could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls. As you can probably guess, many different types of CPUs are used as controllers. The trouble with C and C++ (and most other languages) is that they are designed to be compiled for a specific target. Although it is possible to compile a C++ program for just about any type of CPU, to do so requires a full C++ compiler targeted for that CPU. The problem is that compilers are expensive and time-consuming to create. An easier and more cost efficient solution was needed. In an attempt to find such a solution, Gosling and others began work on a portable, platform-independent language that could be used to produce code that would run on a variety of CPUs under differing environments. This effort ultimately led to the creation of Java [21].

In Fig.3.14 shown screenshot from processing of program. As show in picture we can see four parameters of each three PV modules and other basic

parameters of solar power station. And you can show obtain graphic each connected any parameters.

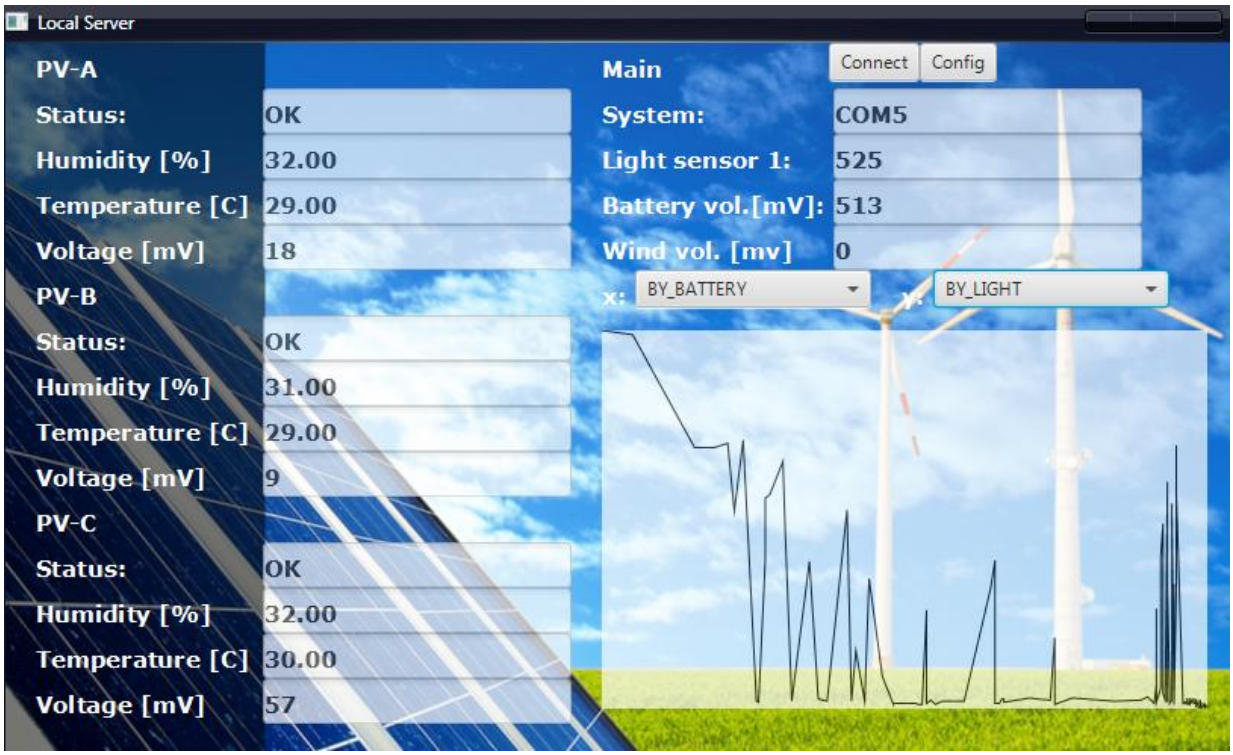

Figure 3.14. Software of the local monitoring system

That time, all information send to another monitoring point, and that point also you obtain graphic each connected any parameters. For example you have not a personal computer or laptop, but you have a mobile phone or tablet which can support android platform. Don’t afraid by using our monitoring system you can obtain any information on your device by using special application, which equipped simple graphic user interface like computer version. The android version of program works wireless by helping Wi-Fi technology.

And last advantage of my autonomous solar power system is using GSM technology. It means if you have not a computer or any device which can support android platform, you can obtain information on your simple mobile phone by using service of SMS.

Conclusion to chapter

1. In this unit we have seen how to connect a autonomous solar power station to the Monitoring Center, how to transmit parameter signals from autonomous solar power stations to the Monitoring Control Center.

2. We learned from this unit how develop this project, how build autonomous solar power stations and at the end how to provide their both work at the same time. Then, I learnt what should do after every steps.

3. In my opinion view while developing this project we will know that it is true and excellent way to use solar power system on the telecommunication objects. Because, this project can provide us be in touch every time. So I can say it is very useful idea. I suggest to develop this project.

4. SAFETY OF VITAL ACTIVITY

4.1. Rational organization of work place

The complexity of production processes and equipment changed the functions of the person in modern industry: increased responsibility of tasks; increased volume of information perceived by the working and the performance of the equipment. A person's work has become more difficult, increased load on the nervous system and increased physical load. In some cases, the man has become the least reliable link of the system «man-machine». There is a task of providing reliability and safety of persons at work. Solves this task ergonomic sand engineering psychology [28].

Ergonomics (from the Greek ergon work and nomos - law) is the scientific discipline that studies the human in terms of its activities related to the use of machines. The goal of ergonomics - optimization of conditions of work in the system "man-machine". Ergonomics defines the requirements of the person to technology and to the conditions of its functioning. The ergonomics of the equipment is the most generalized index of properties and other characteristics of equipment.

The connection of the man with the environment and the parameters of the workplace. Working place, this is the area in which the committed work of the performer or group of performers. Jobs may be individual and collective, universal, specialized and special.

General requirements, which must be observed when designing jobs, the following:

- adequate working space for the person;
- optimum position of the body of the worker;
- sufficient physical, visual and auditory communication between man and machine;
- optimal allocation of working space in the room;

– the permissible level of action of factors of production conditions;
– the optimal placement of the information and the motor field;
– availability of means of protection from hazards.

Design should provide the zone of optimum and easy reach of the motor field of the workplace and the optimal area of the information field of the workplace. Angle of view in relation to the horizontal should be 30-40 degrees.

The choice of working arrangements should take into account the efforts expended by the man, the magnitude of the movements, the need for movement, the pace of operations. The choice of working postures should take into account the physiology of man and parameters of working places determined by the choice of the position of the body at work (standing, sitting, a variable).

Jobs for work «sitting» are organized in an easy job and middle severity, and the severe - working posture -"standing".

In the design of equipment and organization of a job it is necessary to foresee the possibility of regulating the individual elements, in order to ensure the optimum position of the operator.

The design of the equipment must ensure that it meets the anthropometric and bio mechanical characteristics of the individual on the basis of accounting change dynamics of the amount of heat when you move, the range of motion in joints.

For the account in the design of equipment anthropometric data should:
– determine the contingent of people for whom is designed equipment;
– select a group of anthropometric characteristics;
– install the percentage of working, which must meet the equipment;
– determine the boundaries of the interval size (efforts), which should be implemented in the hardware.

When designing the use anthropometric dimensions of the body, and take into account the differences in the sizes of the body of men and women, nationality, age, professional. To determine the boundaries of the intervals, which take account of the percentage of the population, the system is used pertseteley.

Design of the equipment should provide the ability to use at least for 90% of consumers [29].

To work in a position "sitting" are used by various operating seats. Distinguish workers seat for long and short term use. General requirements for the seat of long use of the following: the seat should ensure position, minimizing the statistical work of muscles; create conditions the possibility of changes in working postures; not to obstruct the activities of the systems of the body; to ensure the free movement relative to the working surface, have adjustable parameters; have the floor upholstery. For short-term use is recommended hard chairs and a different type of stools.

In the conditions of growing mechanization and automation of production processes is of special significance means of display of the information about the object of management. Widespread use of the received information model, that is organized according to certain rules information about the status of the object of control.

The information models of the following requirements:

– the content of the information model should adequately display the object of management;

– information model should provide the best information balance;

– the shape and composition of the information of the model must be consistent with the labor process and possibilities of man for the reception of the information.

Practice makes it possible to outline the sequence of the development of an information model: definition of the objectives of the system, the sequence of their decisions and sources of information; drawing up a list of control objects and their characteristics; the distribution of objects on the degree of importance; the distribution of functions between automation and man; the choice of coding of objects and drawing up of the overall composition models; determination of Executive actions of man.

In the process of constructing information model are determined by the location of the media in the workplace, are selected dimensions of marks and the layout of. Displaying means are placed in the field of view of an observer with the account of optimum corners and observation areas. Dimensions signs monitoring are determined taking into account maximum accuracy and speed of perception of the information, as well as the brightness of the character, magnitude contrast, the use of color. Optimum brightness are considered to be the value at which the maximum contrast sensitivity. The value of it will be greater, the smaller the size of the object of discrimination. Optimal area size contrast is 60-90%.

In the work of the eyes is a place of a certain inertia, which requires taking into account the time of exposure of the optic signal and the time intervals for the sense of separate signals the following one after the other. In most cases, the exposure time of the signal should be no less than 50 MS. Each variety of indicators has its area of use: indicators backlit used for the display of high-quality information that requires an immediate response of the operator; gauges are used for the reading of the measured parameters; integral indicators for combining information immediately on several parameters.

The structure and dynamics of the controlled object are usually with the help of a chip. In some cases the scoreboard used to display information and perception of the team of operators.

In the design of the workplace should take into account the rules of the economy's movements: when using two hands of their motion should be simultaneous and balanced; movement should be smooth and rounded, rhythmic and customary for working. The design of the equipment shall take into account the rules relating to the speed and accuracy of workers ' struggles. For example, the most rapid movement to itself; in the horizontal plane of the hand speed more than in the vertical; the accuracy of movements better in a sitting position, than standing, etc. Controls, used in the workplace must comply with the General requirements of ergonomics: and direction of the management bodies must comply with the movement associated with him indicator; the compliance of the location

of the management bodies of the sequence of work of the operator; ease of use; the creation of the bodies of the Board of mechanical resistance and etc. In addition, for each type of bodies of pressure corresponds to a specific area of use and the special requirements of the size, form, effort, etc.

The automated workplace of the operator-Communicator (the operator in the control room) in the General case are used:

- means of mapping the information of individual use (imaging units, signaling devices, and so on);

- means of control and input of information (remote the display, keyboard control, separate controls, and so on);

- devices of communication and transmission of information (modems, telegraphic and telephone sets):

- the device documentation and storage of information (printing devices, magnetic recording and so on);

- auxiliary equipment (means of office equipment, the storage media, the device of local lighting).

At the automated working place should be provided with information and constructive compatibility used by technical means, of anthropometric and physiological characteristics of the person.

At optimization of the procedures of interaction between operators of telecommunications workers with technical means in the conditions of automation ergonomic factors act as the main determining the probability-time characteristics and the intensity of the work. These factors are sensitive to variations of individual properties of the operator.

Working the furniture should be comfortable for the execution of planned operations. The design of the working furniture: table, chairs is of great importance for the creation of healthy environments and highly productive work. Working the furniture is designed with consideration of anthropometric data of a human, technical, aesthetic and economic factors.

In the complete set of the working furniture of great importance is the design of the production of a chair, as it depends on the attitude of the employee and, therefore, energy consumption and the degree of its strain. Operating the seat must have the required dimensions, the relevant anthropometric data of the person and be flexible. The most comfortable chairs and seats with adjustable back tilt and height of seat. Changing the height of the seat from the floor and back angle, you can find the most appropriate labour process and the individual characteristics of the employee.

As a rule, all the surface of the written and desktops should be at the level of the elbow in the position of a person. When choosing the height of the table should be considered a man sits during work or stands.

The inconvenient of the table height reduces the efficiency of work and causes rapid fatigue. The lack of sufficient space for the knees and feet cause constant irritation of the employee. Minimum operating table height should be not less than 725 mm. As practice shows, for the working medium height the height of the desktop is accepted 800 mm. For the employee of another growth you can change the height of the working chair, or the position of the boards so that the distance from the object processing before the eyes of the working height is equal to approximately 450 mm [28].

Accommodation of the technical means and the chair of the operator in the working zone should provide easy access to the main functional nodes and units of equipment for conducting technical diagnostics, preventive inspection and repair; the ability to quickly occupy and to leave the work area; the exception of accidental actuation means of control and input of information; comfortable working posture and position of rest. In addition, the scheme of accommodation should meet the requirements of integrity, compactness and technical and aesthetic expressiveness of the working postures.

The display must be placed on a table or stand so that the distance of observation on the screen does not exceed 700 mm (optimal distance of 450 - 500 mm). Display screen height must be located so that the angle between the centre of

the screen and horizontal line of sight was 200. Horizontal viewing angle of the screen should not exceed 600. The remote display to be placed on a desktop or stand so that the height of the keypad in relation to sex was 650 - 720 mm. When placing the remote control on a standard desktop height of 750 mm it is necessary to use the seat with height adjustable seat (450 - 380 mm) and the footrests.

Document (form) for entry operator data it is recommended to have at a distance of 450 - 500 mm from the eyes of the operator, predominantly on the left, with the angle between display screen and the document in the horizontal plane shall be 30 40 degrees. The tilt angle of the keyboard should be equal to 15 degrees.

Display screen, documents and keypad display should be located so that the difference of brightness surfaces, depending on their location relative to the source of light, not more than 1:10 (the recommended value 1:3). At nominal values of brightness of the image on the screen 50 - 100 CD/m2 illumination of the document should be 300 - 500 Lux.

Working place should be equipped in such a way that the movement of an employee would be the most efficient, least tedious.

The device documentation and other, rarely used by technical means, it is recommended to concentrate on the right from the operator in the zone of maximum reach and means of communication to the left, to free the right hand for the entries.

4.2. Emergencies

In theory SAFETY EMERGENCIES - is a set of events, the result of the onset of which is characterized by one or more of the following signso

a) danger to life and health of a significant number of people;

b) the material violation of the ecological balance in the area of the emergency;

c) the failure of the life support systems and control, full or partial cessation of economic activities;

d) significant material and economic damage;

e) the need to involve large as the usually external to the area of emergency forces and means for the salvation of men and the elimination of consequences;

e) psychological discomfort for large groups of people.

It is characteristic that emergency arises outwardly suddenly, suddenly. Specification of definition of the emergency is achieved by introduction of quantitative measures of the dangers [30].

The classification of emergencies.

For reasons of emergencies are of natural, man-made, man-made, environmental, and social.

To the natural (natural) emergency situations are dangerous natural phenomena or processes that have extraordinary in nature and lead to a breach of everyday life more or less significant groups of the population, loss of life destruction of material values. These include earthquakes, floods, tsunamis, volcanic eruptions, mudflows, landslides, avalanches, hurricanes and Smer-Chi, massive forest and peat fires, snow and avalanches. The number of natural disasters are also droughts, long-term heavy rains, strong stable frosts, epidemics, epizootics, epidemics, mass distribution of pests of agriculture and forestry. Natural disasters can happen: as a result of rapid movement of the substance earthquakes, landslides); in the release of within the earth'senergy (volcanic activity earthquakes) at increasing the overall level of rivers lakes and seas floods tsunamis) under the influence of an unusually strong wind ahurricanes cyclones. Some natural disasters fires avalanches landslides, etc..may arise as a result of the actions of the people themselves but their consequences are always the result of the action of the forces of nature. For each natural disaster characterized by the presence of intrinsic in the affecting factors, adversely affecting human health.

Natural disasters are a tragedy of the entire state and especiallyo for those areas where they occur. As a result of natural disasters are affecting the economy

of the country since the collapse of production of the enterprise the destruction of material values and most importantlyo there are losses among the peoplee killed their housing and property. In addition, natural disasters pose extremely adverse conditions of life for the population, which may be the cause of outbreaks of infectious diseases. The number of people affected by natural disasters can be considerable and the nature of the lesions is very diverse. Most people suffer from floods (40% of the total damage), hurricanes (20%), earthquakes and droughts (15%). About 10% of the total damage is on the other types of disasters [28].

A number of Soviet and foreign experts, citing data on the losses in major disasters assume that in the future in connection with the growth and concentration of population similar in the force of the disaster will be accompanied by an increase in the number of casualties in the tens of times.

Man-made emergency situations is considered a sudden failure of machines, mechanisms and units during their operationи accompanied by serious violations of the production processc the explosions the formation of fire radioactive chemical or biological infections of large territories a group of damage destruction of people. To technogenic emergencies are accidents at industrial facilities constructione as well as on railм airo roado pipeline and water transport as a result of which the firesa the destruction of civil and industrial buildingsa there was a danger of radioactive contaminationя chemical and bacterial contaminationт there was the spreading of the oil products and aggressive poisonous liquid on the surface of earth and water and there are other consequences endangering human health and the environment.

The nature of the consequences of technogenic catastrophes depends on the type of accident, its scale and characteristics of the enterprise, where the crash occurred (on the means of transport and the circumstances in which the accident occurred.

Anthropogenic emergency situations are the consequence of the erroneous actions of the personnel. This class of emergency can occur at the same objects that

and man-made emergency situations. The difference consists only in the fact that man-made emergency situations is not connected with the human factor directly.

The emergency ecological character may include: intensive degradation of the soil and its pollution by heavy metals (cadmium, lead, mercury, chromium, etc.) and other harmful substances, polluting the atmosphere of harmful chemical substances noise electromagnetic fields acid rain the destruction of the ozone layer, etc.

To the social emergency relate the events taking place in the society robbery violence) ethnic conflicts accompanied by the use of force contradictions between the States with the use of weapons.

Conclusion to chapter

A critical benefit is the enhanced safety for personnel as a result of having 24/7 monitor and control centers available to them. When a control center is in use, each service provider is required to check in with the center prior to performing any maintenance. The center can participate in the safety protocols such as “lock out and tag out”, using remote operations and document the personnel entering the area and the activity being performed. When the work is complete, the service provider contacts the center and gives the “all clear” signal. The control center helps close out the “lock out and tag out” process and then documents the completion of work. Upon the safe exit of all personnel, the control center operators restart the equipment and systems remotely. This makes the process—and the service providers—safer by creating a protocol for verification prior to action.

CONCLUSION

1. After this we will know advantages and benefits of solar power system. It is very good way to get energy for free. That's way I decided to use this system on the telecommunication objects. I mean they will work automatically. This project is very enduring. Because it has very quality guaranty.

2. More than 70% of the territory of Uzbekistan is suitable to build and install solar power stations. It means we have more requirement lands for using type of this energy. That is why, we have to build and control work process automatically that stations. For doing this we need to monitoring systems. Local and main monitoring systems are used for the collection information, from any points of system.

3. As the solar industry continues to mature and increase, you can expect solar farms to adopt the same controls and protocols the traditional power plants use across the United States. This change is a welcome sign that the PV solar farms are finally considered mainstream.

4. A program of the local monitoring center should be work fast and correctly. And it demand more speed from local server. For making a program for local or main server firstly we have to choose useful programming language. Java is one of the world's most important and widely used computer languages. Furthermore, it has held that distinction for many years. Java is the most popular and reached with useful classes. That's why I choose Java programming language

THE LIST OF THE USED LITERATURE

1. Доклад Президента Республики Узбекистан Ислама Каримова на заседании Кабинета Министров, посвященном итогам социально-экономического развития в 2014 году и важнейшим приоритетным направлениям экономической программы на 2015 год – 2015.;
2. Speech of Head of the Main Department Ministry of Economy of the Republic of Uzbekistan, 2014y.;
3. Mei Shan Ngan and Chee Wei Tan, "A Study of Maximum Power Point Tracking Algorithms for Stand-alone Photovoltaic Systems", IEEE pplied Power Electronics Colloquium, August 2011.;
4. Dobos, A.P.; An Improved Coefficient Calculator for the CEC 6 Parameter Photovoltaic Module Model. J. Solar Energy Engineering, 134(2), 2012.;
5. Orioli, A. and Di Gangi, A. (2013) A Procedure to Calculate the Five-Parameter Model of Crystalline Silicon Photovoltaic Modules on the Basis of the Tabular Performance Data. Applied Energy, 102, 1160-1177. ;
6. J. Cubas, S. Pindado, M. Victoria, On the analytical approach for modeling photovoltaic systems behavior. J. Power Sources 247, 467^-74 (2014).;
7. H.S. Rauschenbach, Solar Cell Array Design Handbook, The Principles and Technology of Photovoltaic Energy Conversion. (Van Nostrand Reinhold Co, New York, 1980).;
8. Ravi Prakash Tiwari, Rajesh M, K. Sudhakar" Energy and energy analysis of solar photovoltaic system", 2012 Bhopal.;
9. "Natural Forcing of the Climate System". Intergovernmental Panel on Climate Change. Retrieved 2007-9-29.Radiation Budget". NASA Langley Research Center. 2006-10-17.;
10. Enrique, J.M.; Durán E.; Sidrach-de-Cardona M.; Andújar, J.M. Theoretical assessment of the maximum power point tracking efficiency of photovoltaic facilities with different converter topologies. Sol. Energy 2007, 81, 31–38.;

11. Durán, E.; Andújar, J.M.; Segura, F.; Barragán, A.J. A high-flexibility DC load for fuel cell and solar arrays power sources based on DC-DC converters. Appl. Energy 2011, 88, 1690–1702.;
12. Lasseter, R.H., "Extended CERTS microgrid," in Proc. of 2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 20-24 July 2008.;
13. Y. Levron, D. Shmilovitz, "Maximum power point tracking employing sliding mode control", IEEE Transactions on Circuits and Systems, vol. 60, no. 3, pp. 724-732, 2013.;
14. K. Stankovic, M. Vujisic, E. Dolicanin, Reliability Of Semiconductor And Gas-filled Diodes For Over-voltage Protection Exposed To Ionizing Radiation, Nucl. Technol. Radiat., Vol. 24, pp. 132-137, 2009.;
15. N.Marjanovic,M. Vujisic, K. Stankovic, D. Despotovic, P. Osmokrovic, Simulated Exposure Of Titanium DioxideMemristors To Ion Beams, Nucl. Technol. Radiat., Vol. 25, pp. 120-125, 2010.;
16. Y. Cheng, "Assessments of energy capacity and energy losses of supercar-pacitors in fast charging–discharging cycles," IEEE Trans. Energy Convers., vol. 25, no. 1, pp. 253–261, Mar. 2010.;
17. M. Liserre, T. Sauter, and J. Y. Hung, "Future energy systems: Integrating renewable energy sources into the smart power grid through industrial electronics," IEEE Ind. Electron. Mag., vol. 4, no. 1, pp. 18–37, Mar. 2010.;
18. A. El Aroudi andM. Orabi, "Stabilizing technique for AC–DC boost PFC converter based on time delay feedback," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 1, pp. 56–60, Jan. 2010.;
19. Y.A.Hajizadeh,M.A.Golkar, andA. Feliachi, "Voltage control and active power management of hybrid fuel-cell/energy-storage power conversion system under unbalanced voltage sag conditions," IEEE Trans. Energy Convers., vol. 25, no. 4, pp. 1195–1208, Dec. 2010.;
20. P. Thounthong, S. Pierfederici, and B. Davat, "Analysis of differential

flatness-based control for a fuel cell hybrid power source," IEEE Trans. Energy Convers., vol. 25, no. 3, pp. 909–920, Sep. 2010.;

21. Herbert Schildt "Java the complete reference", ninth edition 2014.;
22. stackoverflow.com ;
23. zelectro.cc ;
24. www.phidgets.com;
25. pveducation.org ;
26. www.wikipedia.org ;
27. www.arduino.com;
28. Yormatov G'. Y., Isamuxamedov Y. U. Mehnatni muxofaza qilish Darslik. O'zbekiston Nashriyoti, Toshkent – 2002.;
29. Ecology-Information life safety. Textbook forHigh Schools. MeravyA. 2002.;
30. Socio-Information Broadcasting danger information technologies. Textbookfor High Schools. RysinYS, HeliosARVs, m.:2007.

APPENDIX

THE MINISTRY OF INFORMATION DEVELOPMENT TECHNOLOGY AND COMMUNICATIONS THE REPUBLIC OF UZBEKISTAN
TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

FINAL QUALIFICATION WORK

ON THE THEME:

DEVELOPING THE MODEL FOR MEASURING AND EVALUATING OF PARAMETERS OF THE AUTONOMOUS SOLAR POWER STATION

Graduate: Begmatov Sh. A.
Supervisor: Isayev R.I.

TASHKENT - 2015

Actuality

- Today the energy of solar used widely in every sphere. Because the solar power technology one of the green energy source. But, some defects there are. The photovoltaic technology or PV is a main part of solar panels, which can convert light energy to electrical form. The most outside effects occurs in this point and we should be aware results come by thus effects. Problem is get information about efficient of all objects of solar station or generating electrical power from long distance and process should be work automatically.

APPENDIX

SOFTWARE FOR LOCAL MONITORING SYSTEM
(Java)

SolarEnergy.java

```
package solarenergy;
import java.awt.AWTException;
import java.awt.Event;
import java.awt.MouseInfo;
import java.awt.Point;
import java.io.IOException;
import java.io.InputStream;
import java.net.Socket;
import java.util.Enumeration;
import java.util.Scanner;
import java.util.TooManyListenersException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.comm.*;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundImage;
import javafx.scene.layout.BackgroundPosition;
import javafx.scene.layout.BackgroundRepeat;
import javafx.scene.layout.BackgroundSize;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import static solarenergy.Text.*;

/**
 *
 * @author Shohruh
 */
public class SolarEnergy extends Application implements SerialPortEventListener {
public static String PhoneNumber="";
    static CommPortIdentifier portId;
    static Enumeration portList;
    String Enum = "";
    InputStream inputStream;
    SerialPort serialPort;
    private TextField[] textA = new TextField[4];
    TextField[] textB = new TextField[4];
    TextField[] textC = new TextField[4];
   TextField[] textOther=new TextField[4];
    String[] valueA = new String[4];
    String[] valueB = new String[4];
```

```
    String[] valueC = new String[4];
    static String[] otherValue=new String[4];
    private boolean connectionOpened=false;
    static boolean connectionWiFi=false;
    private StackPane root;
    public static boolean isWorking = true;
    private String str;
    private Network network=new Network();
    private Socket socket;
    public static Values values=new FileManager().read();
    void start() {
        portList = CommPortIdentifier.getPortIdentifiers();

        while (portList.hasMoreElements()) {
            portId = (CommPortIdentifier) portList.nextElement();
            if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
                System.out.println(portId.getName());
                        otherValue[0]=portId.getName();
                openConnection();
                connectionOpened=true;
                break;
            }
        }
    }

    @Override
    public void start(Stage primaryStage) {
otherValue[0]="Off";
        root = new StackPane();
        ShowResultWindow();
        Scene scene = new Scene(root, 820, 500);
        root.setBackground(new Background(new BackgroundImage(new Image("mainBack.png"),
BackgroundRepeat.NO_REPEAT, BackgroundRepeat.REPEAT, BackgroundPosition.DEFAULT,
BackgroundSize.DEFAULT)));
        root.setMaxWidth(800);
        root.setMaxWidth(600);

        primaryStage.setTitle("Local Server");
        primaryStage.setScene(scene);
        primaryStage.setMaxWidth(820);
        primaryStage.setMinWidth(820);
        primaryStage.setMaxHeight(500);
        primaryStage.setMinHeight(500);

        primaryStage.show();
        start();

        new Thread(() -> {
            while (isWorking) {
                try {
                    Platform.runLater(() -> {
                        for (int i = 0; i < 4; i++) {
                            textA[i].setText(valueA[i]);
                            textB[i].setText(valueB[i]);
                            textC[i].setText(valueC[i]);
                          textOther[i].setText(otherValue[i]);
                        }
```

```
                });
                if(!connectionOpened)
                    start();


                Thread.sleep(100);
            } catch (InterruptedException ex) {
            }
        }
    }).start();

    }
public void ConfigurationWindow(){
    root.getChildren().clear();
    TextField field=new TextField();
    field.setOnKeyReleased((event->{
    if(event.getCode().getName().equalsIgnoreCase("Enter"))
    {
        PhoneNumber=field.getText();
        ShowResultWindow();
    }
        }));
    HBox hbox=new HBox(makeLabel("phone numb:"),field);
    root.getChildren().add(hbox);
}
    public void ShowResultWindow() {
root.getChildren().clear();
        Label volLabel = makeLabel("Voltage [mV]");
        Label temLabel = makeLabel("Temperature [C]");
        Label humLabel = makeLabel("Humidity [%]");
        Label PVA = makeLabel("PV-A");
        Label PVB = makeLabel("PV-B");
        Label PVC = makeLabel("PV-C");
        VBox leftLabelBox = new VBox();
        leftLabelBox.getChildren().addAll(volLabel, temLabel, humLabel);
        leftLabelBox.setAlignment(Pos.TOP_RIGHT);

        VBox rightTextA = new VBox();
        HBox hBoxA = new HBox();
        VBox vBoxA = new VBox();

        VBox rightTextB = new VBox();
        HBox hBoxB = new HBox();
        VBox vBoxB = new VBox();

        VBox rightTextC = new VBox();
        HBox hBoxC = new HBox();
        VBox vBoxC = new VBox();

        VBox rightTextOther=new VBox();
        HBox hBoxOther = new HBox();
        VBox vBoxOther = new VBox();

        for (int i = 0; i < 4; i++) {
            textA[i] = makeField();
            textB[i] = makeField();
            textC[i] = makeField();
```

```
            textOther[i]=makeField();
            rightTextA.getChildren().add(textA[i]);
            rightTextB.getChildren().add(textB[i]);
            rightTextC.getChildren().add(textC[i]);
            rightTextOther.getChildren().add(textOther[i]);
        }
        hBoxA.getChildren().addAll(makeRightPanel(), rightTextA);
        vBoxA.getChildren().addAll(PVA, hBoxA);

        hBoxB.getChildren().addAll(makeRightPanel(), rightTextB);
        vBoxA.getChildren().addAll(PVB, hBoxB);

        hBoxC.getChildren().addAll(makeRightPanel(), rightTextC);
        vBoxC.getChildren().addAll(PVC, hBoxC);
        Button btn=new Button("Connect");
          Button btnConfig=new Button("Config");
          btnConfig.setOnMouseClicked((event)->{
             ConfigurationWindow();

          });
        hBoxOther.getChildren().addAll(makeRightPanelLight(), rightTextOther);

        vBoxOther.getChildren().addAll(new
HBox(makeLabel("Main"),btn,btnConfig),hBoxOther,makeCombos(), new Graphic(400, 250).getGrid());


        //192.168.43.1 port 8080

        System.out.println("network status: "+network.isConnected());
       btn.setOnMouseClicked((event)->{
       //System.out.println(new FileManager().write(values));
       network.setConnection("192.168.43.1", 8080);
        });

        VBox vBox2=new VBox(vBoxA,vBoxB,vBoxC);
        HBox hBox = new HBox(vBox2,vBoxOther);
        root.getChildren().add(hBox);
    }
 public static void main(String[] args) {

        launch(args);
        System.out.println("asd");

    }

    @Override
    public void stop() {


        network.sendData("stop");
        isWorking = false;

         try {
             if(inputStream!=null)
            inputStream.close();
        if(serialPort!=null)
        serialPort.close();
```

```
    if(network.isConnected())
    network.closeConnection();
    Thread.sleep(500);
    super.stop();
     } catch (Exception ex) {
         System.out.println("Error");
     }
}

@Override
public void serialEvent(SerialPortEvent event) {
    if(connectionOpened)
    switch (event.getEventType()) {
        case SerialPortEvent.BI:
        case SerialPortEvent.OE:
        case SerialPortEvent.FE:
        case SerialPortEvent.PE:
        case SerialPortEvent.CD:
        case SerialPortEvent.CTS:
        case SerialPortEvent.DSR:
        case SerialPortEvent.RI:
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
            break;
        case SerialPortEvent.DATA_AVAILABLE:

            Scanner scan = new Scanner(inputStream);

            while (scan.hasNext() && isWorking) {
                str = scan.next();
                if (str != null && str.length() > 5) {
                    String subStr = str.substring(0, 5);
                    switch (subStr) {
                        case "A->M:":
                            valueA[0] = str.substring(5);
                            break;
                        case "A->H:":
                            valueA[1] = str.substring(5);
                            break;
                        case "A->T:":
                            valueA[2] = str.substring(5);
                            break;
                        case "A->V:":
                            valueA[3] = str.substring(5);
                            break;
                        case "B->M:":
                            valueB[0] = str.substring(5);
                            break;
                        case "B->H:":
                            valueB[1] = str.substring(5);
                            break;
                        case "B->T:":
                            valueB[2] = str.substring(5);
                            break;
                        case "B->V:":
                            valueB[3] = str.substring(5);
                            break;
                        case "C->M:":
```

```
                    valueC[0] = str.substring(5);
                    break;
                case "C->H:":
                    valueC[1] = str.substring(5);
                    break;
                case "C->T:":
                    valueC[2] = str.substring(5);
                    break;
                case "C->V:":
                    valueC[3] = str.substring(5);
                    break;
                case "C->L:":
                    otherValue[1] = str.substring(5);
                    break;
                case "B->L:":
                    otherValue[2] = str.substring(5);
                    break;
                case "A->L:":
                    otherValue[3] = str.substring(5);
                    break;
            }
            values.addValue(str);
            if(connectionWiFi)
             network.sendData(str);

        }
      }
      break;
   }
}

private void openConnection() {
   try {
      serialPort = (SerialPort) portId.open("SimpleReadApp", 2000);
   } catch (PortInUseException e) {
      System.out.println(e);
   }
   try {
      inputStream = serialPort.getInputStream();
   } catch (IOException e) {
      System.out.println(e);
   }
   try {
      serialPort.addEventListener(this);
   } catch (TooManyListenersException e) {
      System.out.println(e);
   }
   serialPort.notifyOnDataAvailable(true);
   try {
      serialPort.setSerialPortParams(9600,
            SerialPort.DATABITS_8,
            SerialPort.STOPBITS_1,
            SerialPort.PARITY_NONE);
   } catch (UnsupportedCommOperationException e) {
      System.out.println(e);
   }
}
```

```
}
```

Values.java

```
package solarenergy;

import java.util.ArrayList;

/**
 *
 * @author Shohruh
 */
public class Values {

   private ArrayList<Value> value = new ArrayList<Value>();

   boolean isNewValue = true;
   Value workerValue;

   public Values() {
      workerValue = new Value();
   }
   public Value at(int index){
      return value.get(index);
   }
   public ArrayList<Value> sortBy(NewEnum Type) {
      for (int i = 0; i < value.size(); i++) {
         for (int j = i + 1; j < value.size(); j++) {
           try{if (value.get(i).get(Type) > value.get(j).get(Type)) {
               value.get(i).exchangeValue(value.get(j));
            }}
           catch(Exception e){
                 e.printStackTrace();

           }

         }
      }
      return value;
   }
public int size(){
   return value.size();
}
private boolean findClone(){
   for(Value val:value){
         if(val.isSame(workerValue)){
            return true;}
   }
   return false;

}
   private double setValue(String str) {

      if (workerValue.isFilled()) {
          if(!findClone()) value.add(workerValue);
      workerValue = new Value();
      }
```

```
    try{
    return Double.valueOf(str.substring(5));}
    catch(Exception e){
       System.out.println(str);
    return 0;
    }
  }

  public void addValue(String value) {
    if (value.length() > 5) {
       switch (value.substring(0, 5)) {
          case "A->V:":
             workerValue.voltageA = setValue(value);
             break;
          case "A->H:":
             workerValue.humA = setValue(value);
             break;
          case "A->T:":
             workerValue.tempA = setValue(value);
             break;
          case "B->V:":
             workerValue.voltageB = setValue(value);
             break;
          case "B->H:":
             workerValue.humB = setValue(value);
             break;
          case "B->T:":
             workerValue.tempB = setValue(value);
             break;
          case "C->V:":
             workerValue.voltageC = setValue(value);
             break;
          case "C->H:":
             workerValue.humC = setValue(value);
             break;
          case "C->T:":
             workerValue.tempC = setValue(value);
             break;
          case "A->L:":
             workerValue.wind = setValue(value);
             break;
          case "B->L:":
             workerValue.light = setValue(value);
             break;
          case "C->L:":
             workerValue.battery = setValue(value);
             break;

       }
      }
  }
   private void addValue(int voltageA, int voltageB, int voltageC, int humA, int humB, int humC, int
tempA, int tempB, int tempC, int wind, int battery, int light) {
      value.add(new Value(voltageA, voltageB, voltageC, humA, humB, humC, tempA, tempB, tempC,
wind, battery, light));
   }}
```

Network.java

```
package solarenergy;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import static solarenergy.SolarEnergy.otherValue;
import static solarenergy.SolarEnergy.connectionWiFi;
import static solarenergy.SolarEnergy.*;

/**
 *
 * @author Shohruh
 */
public class Network {
int count=0;
    //192.168.43.1 port 8080
    Socket socket;
    InputStream is;
    OutputStream os;
    Scanner scan;
    PrintWriter pw;
    boolean isWorking = true;
    boolean pvAWorks = true, pvBWorks = true, pvCWorks = true;

    public boolean isConnected() {
        if (socket == null) {
            return false;
        }
        return true;
    }

    public void start() {

        System.out.println("I'm listening");
        try {
            os = socket.getOutputStream();
        } catch (IOException ex) {
            Logger.getLogger(Network.class.getName()).log(Level.SEVERE, null, ex);
        }
        pw = new PrintWriter(os);
        sendData("S->M:" + otherValue[0]);
        new Thread(() -> {
            while (isWorking) {
                for (int i = 0; i < 4; i++) {
                    sendData(otherValue[i]);

                }
```

```
            System.out.println("data sent");
        }
    });

}

public void sendData(String str) {
    pw.println(str);
    pw.flush();
    if (str.contains("Time") && PhoneNumber.length() > 7) {
        if (str.contains("A->") && pvAWorks) {
            pvAWorks = false; sendData("SMS:" + PhoneNumber + str);

        } else if (str.contains("B->") && pvBWorks) {
            pvBWorks = false;sendData("SMS:" + PhoneNumber + str);

        } else if (str.contains("C->") && pvCWorks) {
            pvCWorks = false;
            System.out.println("pv c: false");
            sendData("SMS:" + PhoneNumber + str);
        }

    } else if (str.contains("OK") && PhoneNumber.length() > 7) {
        if (!pvAWorks&&str.contains("A->"))
            {   pvAWorks = false;
                sendData("SMS:" + PhoneNumber + str);
                pvAWorks = true;
            }
            else if (!pvBWorks&&str.contains("B->"))
            {  pvBWorks = true;
                sendData("SMS:" + PhoneNumber + str);
              }
         else    if (!pvCWorks&&str.contains("C->"))
            {pvCWorks = true;
                sendData("SMS:" + PhoneNumber + str);

                    System.out.println("pv c: ok");
            }
        }
    }

public boolean setConnection(String add, int port) {
    try {
        socket = new Socket("192.168.43.1", port);
        connectionWiFi = true;
    } catch (IOException ex) {
        System.out.println("socket ochilmadi");
        return false;
    }
    start();
    return true;
}

public void closeConnection() {
    try {
        isWorking = false;
```

```
            os.close();
            socket.close();
        } catch (IOException ex) {
            Logger.getLogger(Network.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
}
```

Graphic.java

```
package solarenergy;

import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Paint;
import static solarenergy.SolarEnergy.values;
import static solarenergy.Value.*;
import static solarenergy.SolarEnergy.isWorking;

/**
 *
 * @author Shohruh
 */
public class Graphic {

    public static NewEnum type1 = NewEnum.BY_VOLTAGE_A, type2 =
NewEnum.BY_TEMPERATURE_A;
    double maxX, maxY, minX, minY,stepX,stepY;
    Canvas canvas;int border=20;
    GraphicsContext gc;
     double []pointX,pointY;
     private double width,height;

    public Graphic(double width, double height) {
        canvas = new Canvas(this.width=width,this.height=height);
        canvas.setOpacity(0.7);

         gc = canvas.getGraphicsContext2D();
        new Thread(() -> {
            while (isWorking) {
                Platform.runLater(() -> {
                    draw();
                });
                try {
                    Thread.sleep(100);
                } catch (InterruptedException ex) {
                    Logger.getLogger(Graphic.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }).start();    }
```

```
    public Canvas getCanvas() {
        return canvas;
    }
    public GridPane getGrid(){
        GridPane gp=new GridPane();
        gp.add(canvas,0,0);
        gp.setPadding(new Insets(10,0,0,20));
        return gp;
    }

    private void draw() {
        gc.setFill(Paint.valueOf("white"));
        gc.fillRect(0, 0, canvas.getWidth(), canvas.getHeight());
        values.sortBy(type1);
        findMaxMin();
        for (int i = 0; i < values.size(); i++) {
            setPoint(values.at(i),i);
        }
        gc.strokePolyline(pointX,pointY,values.size());

    }
private void setPoint(Value val1,int i){
    pointX[i]=(val1.get(type1)-minX)*stepX+border;
    pointY[i]=(val1.get(type2)-minY)*stepY+border;
}
    private void drawLine(Value val1, Value val2) {

            gc.strokeLine((val1.get(type1)-minX)*stepX, (val1.get(type2)-minY)*stepY, (val2.get(type1)-
minX)*stepX, (val2.get(type2)-minY)*stepY);

    }

    private void findMaxMin() {
        pointX=new double[values.size()];
        pointY=new double[values.size()];

        maxX = Double.MIN_VALUE;
        maxY = maxX;
        minX = Double.MAX_VALUE;
        minY = minX;
        for (int i = 0; i < values.size(); i++) {
            if (maxX < values.at(i).get(type1)) {
                maxX = values.at(i).get(type1);
            }
            if (minX > values.at(i).get(type1)) {
                minX = values.at(i).get(type1);
            }

            if (maxY < values.at(i).get(type2)) {
                maxY = values.at(i).get(type2);
            }
            if (minY > values.at(i).get(type2)) {
                minY = values.at(i).get(type2);
            }

        }
        gc.setFill(Paint.valueOf("red"));
```

```
        gc.fillText(String.valueOf(maxX), width-border-10, height-border/2);
        gc.fillText(String.valueOf(minX), border, height-border/2);
        gc.fillText(String.valueOf(maxY), 0, border);
        gc.fillText(String.valueOf(minY), 0, height-border-10);
        stepX=(width-border*2)/(maxX-minX);
        stepY=(height-border*2)/(maxY-minY);

    }

}
```

FileManager.java

```
package solarenergy;

import java.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Shohruh
 */
public class FileManager {

    public FileManager() {
    }
public Values read(){
        ObjectInputStream ois=null;
        Values val=null;
    try {
            FileInputStream fis=new FileInputStream("temp.data");
            ois=new ObjectInputStream(fis);
            val=(Values)ois.readObject();
            System.out.println("read");
            return val;
        } catch (Exception ex) {
            System.out.println("exception");
            return new Values();
        }

}
    public boolean write(Values obj) {
        ObjectOutputStream oos=null;
    FileOutputStream fos;

        try {
            fos = new FileOutputStream("temp.data",true);
            oos = new ObjectOutputStream(fos);
            oos.writeObject(obj);
            System.out.println("written");
        } catch (Exception ex) {
            return false;
        } finally{
        if(oos!=null)
            try {
```

```
            oos.close();
        } catch (IOException ex) {
            System.out.println("exception");
        }
        }

        return true;
    }
}
```

Value.java

```
package solarenergy;

/**
 *
 * @author Shohruh
 */
public class Value {

    static final int BY_VOLTAGE_A = 0;
    static final int BY_VOLTAGE_B = 1;
    static final int BY_VOLTAGE_C = 2;
    static final int BY_HUMIDITY_A = 3;
    static final int BY_HUMIDITY_B = 4;
    static final int BY_HUMIDITY_C = 5;
    static final int BY_TEMPERATURE_A = 6;
    static final int BY_TEMPERATURE_B = 7;
    static final int BY_TEMPERATURE_C = 8;
    static final int BY_LIGHT = 9;
    static final int BY_BATTERY = 10;
    static final int BY_WIND = 11;
    double voltageA = Integer.MAX_VALUE;
    double voltageB = Integer.MAX_VALUE;
    double voltageC = Integer.MAX_VALUE;
    double humA = Integer.MAX_VALUE;
    double humB = Integer.MAX_VALUE;
    double humC = Integer.MAX_VALUE;
    double tempA = Integer.MAX_VALUE;
    double tempB = Integer.MAX_VALUE;
    double tempC = Integer.MAX_VALUE;
    double wind = Integer.MAX_VALUE;
    double battery = Integer.MAX_VALUE;
    double light = Integer.MAX_VALUE;

    @Override
    public Value clone() {
        return new Value(voltageA, voltageB, voltageC, humA, humB, humC, tempA, tempB, tempC, wind,
battery, light);

    }

    public void exchangeValue(Value value) {
        Value val2 = value.clone();
         getValue(value);
        setValue(val2);
```

```
}

private void setValue(Value value) {
   this.voltageA = value.voltageA;
   this.voltageB = value.voltageB;
   this.voltageC = value.voltageC;
   this.humA = value.humA;
   this.humB = value.humB;
   this.humC = value.humC;
   this.tempA = value.tempA;
   this.tempB = value.tempB;
   this.tempC = value.tempC;
   this.wind = value.wind;
   this.battery = value.battery;
   this.light = value.light;
}

private void getValue(Value value) {

   value.voltageA = voltageA;
   value.voltageB = voltageB;
   value.voltageC = voltageC;
   value.humA = humA;
   value.humB = humB;
   value.humC = humC;
   value.tempA = tempA;
   value.tempB = tempB;
   value.tempC = tempC;
   value.wind = wind;
   value.battery = battery;
   value.light = light;

}
public boolean isSame(Value val){
    if (voltageA != val.voltageA) {
      return false;
   }
   if (voltageB != val.voltageB) {
      return false;
   }
   if (voltageC != val.voltageC) {
      return false;
   }
   if (humA != val.humA) {
      return false;
   }
   if (humB != val.humB) {
      return false;
   }
   if (humC != val.humC) {
      return false;
   }
   if (tempA != val.tempA) {
      return false;
   }
   if (tempB != val.tempB) {
```

```
            return false;
        }
        if (tempC != val.tempC) {
            return false;
        }
        if (wind != val.wind) {
            return false;
        }
        if (battery != val.battery) {
            return false;
        }
        if (light !=val.light) {
            return false;
        }
        return true;

    }
public double get(NewEnum newEnum){
    switch (newEnum) {
            case BY_VOLTAGE_A:
                return voltageA;
            case BY_VOLTAGE_B:
                return voltageB;
            case BY_VOLTAGE_C:
                return voltageC;
            case BY_HUMIDITY_A:
                return humA;
            case BY_HUMIDITY_B:
                return humB;
            case BY_HUMIDITY_C:
                return humC;
            case BY_TEMPERATURE_A:
                return tempA;
            case BY_TEMPERATURE_B:
                return tempB;
            case BY_TEMPERATURE_C:
                return tempC;
            case BY_LIGHT:
                return light;
            case BY_BATTERY:
                return battery;
            case BY_WIND:
                return wind;
            default:
                return Integer.MIN_VALUE;
        }
}
    public double get(int index) {
        switch (index) {
            case BY_VOLTAGE_A:
                return voltageA;
            case BY_VOLTAGE_B:
                return voltageB;
            case BY_VOLTAGE_C:
                return voltageC;
            case BY_HUMIDITY_A:
                return humA;
```

```
            case BY_HUMIDITY_B:
                return humB;
            case BY_HUMIDITY_C:
                return humC;
            case BY_TEMPERATURE_A:
                return tempA;
            case BY_TEMPERATURE_B:
                return tempB;
            case BY_TEMPERATURE_C:
                return tempC;
            case BY_LIGHT:
                return light;
            case BY_BATTERY:
                return battery;
            case BY_WIND:
                return wind;
            default:
                return Integer.MIN_VALUE;
        }
    }

    public boolean isFilled() {
        if (voltageA == Integer.MAX_VALUE) {
            return false;
        }
        if (voltageB == Integer.MAX_VALUE) {
            return false;
        }
        if (voltageC == Integer.MAX_VALUE) {
            return false;
        }
        if (humA == Integer.MAX_VALUE) {
            return false;
        }
        if (humB == Integer.MAX_VALUE) {
            return false;
        }
        if (humC == Integer.MAX_VALUE) {
            return false;
        }
        if (tempA == Integer.MAX_VALUE) {
            return false;
        }
        if (tempB == Integer.MAX_VALUE) {
            return false;
        }
        if (tempC == Integer.MAX_VALUE) {
            return false;
        }
        if (wind == Integer.MAX_VALUE) {
            return false;
        }
        if (battery == Integer.MAX_VALUE) {
            return false;
        }
        if (light == Integer.MAX_VALUE) {
            return false;
```

```
        }
        return true;

    }

    public Value() {

    }

    public Value(double voltageA, double voltageB, double voltageC, double humA, double humB, double humC, double tempA, double tempB, double tempC, double wind, double battery, double light) {

        this.voltageA = voltageA;
        this.voltageB = voltageB;
        this.voltageC = voltageC;
        this.humA = humA;
        this.humB = humB;
        this.humC = humC;
        this.tempA = tempA;
        this.tempB = tempB;
        this.tempC = tempC;
        this.wind = wind;
        this.battery = battery;
        this.light = light;

    }

}
```

Text.java

```
package solarenergy;

import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.effect.DropShadow;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.ContextMenuEvent;
import javafx.scene.layout.Border;
import javafx.scene.layout.BorderImage;
import javafx.scene.layout.BorderRepeat;
import javafx.scene.layout.BorderWidths;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.TextAlignment;
import static solarenergy.NewEnum.*;
import static solarenergy.Graphic.*;

/**
```

```
 *
 * @author Shohruh
 */
public class Text {

    public static TextField makeField() {
        TextField txt = new TextField("");
        txt.setMinHeight(30);

        txt.setPadding(new Insets(5, 0, 0, 0));
        txt.setFont(Font.font("verdana", FontWeight.BOLD, 15));
        txt.setOpacity(0.7);
        txt.setEditable(false);

        return txt;
    }

    public static Label makeLabel(String str) {
        Label lbl = new Label(str);
        lbl.setMinHeight(30);
        lbl.setMinWidth(170);
        lbl.setTextFill(Color.WHITE);
        lbl.setFont(Font.font("verdana", FontWeight.BOLD, 15));
        lbl.setTextAlignment(TextAlignment.RIGHT);
        lbl.setPadding(new Insets(5, 0, 0, 20));

            return lbl;
    }
    public static Label makeLabel(String str,boolean b) {
        Label lbl = new Label(str);
        lbl.setMinHeight(30);
        lbl.setMinWidth(40);

        lbl.setTextFill(Color.WHITE);
        lbl.setFont(Font.font("verdana", FontWeight.BOLD, 15));
        lbl.setTextAlignment(TextAlignment.RIGHT);
        lbl.setPadding(new Insets(5, 0, 0, 20));

        return lbl;
    }

    public static VBox makeRightPanel() {
        Label status = makeLabel("Status: ");
        Label volLabel = makeLabel("Voltage [mV] ");
        Label temLabel = makeLabel("Temperature [C] ");
        Label humLabel = makeLabel("Humidity [%] ");
        return new VBox(status, humLabel, temLabel, volLabel);
    }

    public static VBox makeRightPanelLight() {
        Label system = makeLabel("System: ");
        Label lSensor1 = makeLabel("Light sensor 1: ");
        Label lSensor2 = makeLabel("Battery vol.[mV]: ");
        Label windVoltage = makeLabel("Wind vol. [mv]");
        return new VBox(system, lSensor1, lSensor2, windVoltage);
    }
```

```
    public static HBox makeCombos() {
        ComboBox box1 = makeComboBox(BY_VOLTAGE_A);

        box1.setOnHidden((event) -> {
            type1 = (NewEnum) box1.getValue();
        });
        ComboBox box2 = makeComboBox(BY_TEMPERATURE_A);
        box2.setOnHidden((event) -> {
            type2 = (NewEnum) box2.getValue();
        });

        return new HBox(makeLabel("x: ",true), box1, makeLabel("y: ",true), box2);
    }

    public static ComboBox makeComboBox(NewEnum Enum) {

        ComboBox box = new ComboBox();
        box.getItems().addAll(BY_VOLTAGE_A,
                BY_VOLTAGE_B,
                BY_VOLTAGE_C,
                BY_HUMIDITY_A,
                BY_HUMIDITY_B,
                BY_HUMIDITY_C,
                BY_TEMPERATURE_A,
                BY_TEMPERATURE_B,
                BY_TEMPERATURE_C,
                BY_LIGHT,
                BY_BATTERY,
                BY_WIND);
        box.setValue(Enum);

        return box;

    }
}
```

NewEnum.java

```
package solarenergy;
public enum NewEnum {

    BY_VOLTAGE_A,
    BY_VOLTAGE_B,
    BY_VOLTAGE_C,
    BY_HUMIDITY_A,
    BY_HUMIDITY_B,
    BY_HUMIDITY_C,
    BY_TEMPERATURE_A,
    BY_TEMPERATURE_B,
    BY_TEMPERATURE_C,
    BY_LIGHT,
    BY_BATTERY,
    BY_WIND;

}
```

SOFTWARE FOR ANOTHER MONITORING SYSTEM

(Android)

MainActivity.java

```
package com.example.mobilemonitoring;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.WorkSource;
import android.app.Activity;
import android.database.DataSetObserver;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.Menu;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.MotionEvent;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnTouchListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.SpinnerAdapter;
import android.widget.TextView;
import android.widget.Toast;
import static com.example.mobilemonitoring.SMS.*;
public class MainActivity extends Activity {
          public static Parameter type1 = Parameter.BY_VOLTAGE_A, type2 =
Parameter.BY_TEMPERATURE_A;

        public static boolean isWorks = true;
public static double width,height;
        public ServerSocket serverSocket;
        SurfaceView surfaceView;
        Spinner spinX, spinY;
        Canvas canvas;
        public static SurfaceHolder holder;
public static Values values;
```

```
@Override
protected void onPause() {
	try{

		isWorks = false;
	}
	catch(Exception e){

	}

	super.onPause();
}
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		spinX = (Spinner) findViewById(R.id.spinX);
		values=new Values();
		ArrayAdapter adapter = new ArrayAdapter(this,
				android.R.layout.simple_spinner_dropdown_item,
				Parameter.values());
		spinX.setAdapter(adapter);

		spinY = (Spinner) findViewById(R.id.spinY);
		spinY.setAdapter(adapter);
		surfaceView = (SurfaceView) findViewById(R.id.surface);

		surfaceView.getHolder().addCallback(new Callback() {

			@Override
			public void surfaceDestroyed(SurfaceHolder arg0) {

			}

			@Override
			public void surfaceCreated(SurfaceHolder arg0) {
				new Graphic().start();

			}

			@Override
			public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2,
					int arg3) {

			}
		});

		holder = surfaceView.getHolder();

		spinY.setOnItemSelectedListener(new OnItemSelectedListener() {

			@Override
			public void onItemSelected(AdapterView<?> arg0, View arg1,
					int arg2, long arg3) {
				type2=(Parameter)arg0.getSelectedItem();
```

```
                }

                @Override
                public void onNothingSelected(AdapterView<?> arg0) {
                }
        });
        spinX.setOnItemSelectedListener(new OnItemSelectedListener() {

                @Override
                public void onItemSelected(AdapterView<?> arg0, View arg1,
                                int arg2, long arg3) {
                type1=(Parameter) arg0.getSelectedItem();
                }

                @Override
                public void onNothingSelected(AdapterView<?> arg0) {

                }
        });
        try {
                serverSocket = new ServerSocket(8080);
                new Thread(new Runnable() {

                        @Override
                        public void run() {
                                Socket socket;
                                try {
                                        socket = serverSocket.accept();
                                        new myClient(socket).execute();

                                } catch (IOException e) {
                                        e.printStackTrace();
                                }

                        }
                }).start();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}

@Override
protected void onStop() {
        try{
                isWorks = false;
        }
        catch(Exception e){

        }
        super.onStop();
}

void response(Socket socket) {
        try {
                OutputStream os = socket.getOutputStream();
                PrintWriter pw = new PrintWriter(os);
```

```
                pw.write("hy write");
                pw.print("hy print");
                pw.flush();

        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
        return true;
}

class myClient extends AsyncTask<Void, String, Void> {

        InputStream is;
        Socket socket;
        Scanner scan;

        myClient(Socket socket) {
                try {
                        this.socket = socket;
                        is = socket.getInputStream();
                        scan = new Scanner(is);
                } catch (IOException e) {

                        e.printStackTrace();
                }
        }

        @Override
        protected Void doInBackground(Void... arg0) {
                publishProgress("receviving");
                String str="";
                while (isWorks && !socket.isClosed()) {
                        str=scan.nextLine();
                        publishProgress(str);
                values.addValue(str);
                }

                try {if(is!=null)
                        is.close();
                if(socket!=null)
                        socket.close();
                } catch (IOException e) {
                e.printStackTrace();
                }

                return null;
        }

        @Override
        protected void onProgressUpdate(String... values) {
                String str = values[0];
```

```
if (str.startsWith("A->M:")) {
        ((TextView) findViewById(R.id.statusA)).setText(str
                .substring(5));
} else if (str.startsWith("A->H:")) {
        ((TextView) findViewById(R.id.humidityA)).setText(str
                .substring(5));
} else if (str.startsWith("A->T:")) {
        ((TextView) findViewById(R.id.temperatureA)).setText(str
                .substring(5));
} else if (str.startsWith("A->V:")) {
        ((TextView) findViewById(R.id.voltageA)).setText(str
                .substring(5));
} else if (str.startsWith("A->L:")) {
        ((TextView) findViewById(R.id.windErgy)).setText(str
                .substring(5));
} else if (str.startsWith("S->M:")) {
        ((TextView) findViewById(R.id.system))
                .setText(str.substring(5));
} else if (str.startsWith("B->M:")) {
        ((TextView) findViewById(R.id.statusB)).setText(str
                .substring(5));
} else if (str.startsWith("B->H:")) {
        ((TextView) findViewById(R.id.humidityB)).setText(str
                .substring(5));
} else if (str.startsWith("B->T:")) {
        ((TextView) findViewById(R.id.temperatureB)).setText(str
                .substring(5));
} else if (str.startsWith("B->V:")) {
        ((TextView) findViewById(R.id.voltageB)).setText(str
                .substring(5));
} else if (str.startsWith("B->L:")) {
        ((TextView) findViewById(R.id.voltageBat)).setText(str
                .substring(5));
}

else if (str.startsWith("C->M:")) {
        ((TextView) findViewById(R.id.statusC)).setText(str
                .substring(5));
} else if (str.startsWith("C->H:")) {
        ((TextView) findViewById(R.id.humidityC)).setText(str
                .substring(5));
} else if (str.startsWith("C->T:")) {
        ((TextView) findViewById(R.id.temperatureC)).setText(str
                .substring(5));
} else if (str.startsWith("C->V:")) {
        ((TextView) findViewById(R.id.voltageC)).setText(str
                .substring(5));
} else if (str.startsWith("C->L:")) {
        ((TextView) findViewById(R.id.lightSensor)).setText(str
                .substring(5));
} else if (str.startsWith("stop")) {
        try {
                socket.close();
        } catch (IOException e) {
                e.printStackTrace();
        }
```

```
                }
                super.onProgressUpdate(values);
            }

        }

}
```

res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/panel"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="20dp"
  android:paddingRight="20dp"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity" >

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="horizontal"
      android:weightSum="3" >

      <LinearLayout
         android:layout_width="0dp"
         android:layout_height="match_parent"
         android:layout_weight="1"
         android:orientation="vertical"
         android:weightSum="2" >

         <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:orientation="vertical"
            android:weightSum="5" >

            <TextView
               style="@style/text"
               android:layout_width="match_parent"
               android:layout_height="0dp"
               android:layout_weight="1"
               android:gravity="center"
               android:text="@string/PVA"
               android:textAlignment="gravity" />

            <LinearLayout
               android:layout_width="match_parent"
               android:layout_height="0dp"
               android:layout_weight="1"
               android:orientation="horizontal"
               android:weightSum="2" >
```

```
    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/Status"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/statusA"
        style="@style/EditText"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:enabled="false" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >

    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/Humidity"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/humidityA"
        style="@style/EditText"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:enabled="false" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >

    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/Temperature"
        android:textAlignment="center" />
```

```
        <TextView
            android:id="@+id/temperatureA"
            style="@style/EditText"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:enabled="false" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:weightSum="2" >

        <TextView
            style="@style/text"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="@string/voltage"
            android:textAlignment="center" />

        <TextView
            android:id="@+id/voltageA"
            style="@style/EditText"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:enabled="false" />
    </LinearLayout>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="vertical"
    android:weightSum="5" >

    <TextView
        style="@style/text"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="@string/Main"
        android:textAlignment="gravity" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:weightSum="2" >

        <TextView
```

```
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/System"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/system"
        style="@style/EditText"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:enabled="false" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >

    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/LigthSenor"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/lightSensor"
        style="@style/EditText"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:enabled="false" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >

    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/VoltageBat"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/voltageBat"
```

```
            style="@style/EditText"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:enabled="false" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:orientation="horizontal"
            android:weightSum="2" >

            <TextView
                style="@style/text"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="@string/WindEnergy"
                android:textAlignment="center" />

            <TextView
                android:id="@+id/windErgy"
                style="@style/EditText"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:enabled="false" />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>

<LinearLayout
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="2"
    android:orientation="vertical"
    android:weightSum="2" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:weightSum="2" >

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:orientation="vertical"
            android:weightSum="5" >

            <TextView
                style="@style/text"
                android:layout_width="match_parent"
                android:layout_height="0dp"
```

```
    android:layout_weight="1"
    android:text="@string/PVB"
    android:textAlignment="gravity" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >

    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/Status"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/statusB"
        style="@style/EditText"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:enabled="false" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >

    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/Humidity"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/humidityB"
        style="@style/EditText"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:enabled="false" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
```

```
        android:weightSum="2" >

        <TextView
            style="@style/text"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="@string/Temperature"
            android:textAlignment="center" />

        <TextView
            android:id="@+id/temperatureB"
            style="@style/EditText"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:enabled="false" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:weightSum="2" >

        <TextView
            style="@style/text"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="@string/voltage"
            android:textAlignment="center" />

        <TextView
            android:id="@+id/voltageB"
            style="@style/EditText"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:enabled="false" />
    </LinearLayout>
</LinearLayout>

<LinearLayout
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="vertical"
    android:weightSum="5" >

    <TextView
        style="@style/text"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="@string/PVC"
```

```
    android:textAlignment="gravity" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >

    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/Status"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/statusC"
        style="@style/EditText"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:enabled="false" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >

    <TextView
        style="@style/text"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="@string/Humidity"
        android:textAlignment="center" />

    <TextView
        android:id="@+id/humidityC"
        style="@style/EditText"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:enabled="false" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:weightSum="2" >
```

```
            <TextView
                style="@style/text"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="@string/Temperature"
                android:textAlignment="center" />

            <TextView
                android:id="@+id/temperatureC"
                style="@style/EditText"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:enabled="false" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:orientation="horizontal"
            android:weightSum="2" >

            <TextView
                style="@style/text"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="@string/voltage"
                android:textAlignment="center" />

            <TextView
                android:id="@+id/voltageC"
                style="@style/EditText"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:enabled="false" />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="vertical"
    android:weightSum="8" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:weightSum="8" >
```

```
            <TextView
               style="@style/text"
               android:layout_width="0dp"
               android:layout_height="match_parent"
               android:layout_weight="1"
               android:text=" X: " />

            <Spinner
               android:id="@+id/spinX"
               android:layout_width="0dp"
               android:layout_height="match_parent"
               android:layout_weight="3" />

            <TextView
               style="@style/text"
               android:layout_width="0dp"
               android:layout_height="match_parent"
               android:layout_weight="1"
               android:text=" Y: " />

            <Spinner
               android:id="@+id/spinY"
               android:layout_width="0dp"
               android:layout_height="match_parent"
               android:layout_weight="3" />
         </LinearLayout>

         <SurfaceView
            android:id="@+id/surface"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="7"

          />
       </LinearLayout>
    </LinearLayout>

</RelativeLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.mobilemonitoring"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="18" />
         <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
```

```
    <application

        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:screenOrientation="landscape"
            android:name="com.example.mobilemonitoring.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">MobileMonitoring</string>
    <string name="action_settings">Settings</string>
    <string name="PVA">PV A</string>
    <string name="voltage">Voltage [mV]</string>
    <string name="Humidity">Humidity [%]</string>
    <string name="Temperature">Temp. [C] </string>
    <string name="Status">Status :</string>
    <string name="PVB">PV B</string>
    <string name="PVC">PV C</string>
    <string name="Main">Main</string>
    <string name="WindEnergy">Wind Energy [mV]</string>
   <string name="LigthSenor">Light Senson</string>
     <string name="VoltageBat">Battery vol. [mV]</string>
       <string name="System">System</string>
</resources>
```

res/values/styles.xml

```
<resources xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools">

    <style name="text" >
      <item name="android:background"> @android:color/background_dark</item>
       <item name="android:alpha" tools:targetApi="11">0.5</item>

       <item name="android:textStyle">bold</item>
       <item name="android:textColor">@android:color/background_light</item>
                    <item name="android:textAlignment" tools:targetApi="17">textEnd</item>
        <item  name="android:gravity">center</item>
    </style>
<style name="EditText">
```

```
   <item name="android:background">@android:color/white</item>
   <item name="android:textStyle">italic</item>
   <item name="android:textColor"> @android:color/background_dark</item>
      <item name="android:alpha" tools:targetApi="11">0.5</item>
</style>
<style name="surface">

   <item name="android:alpha" tools:targetApi="11">0.7</item>
</style>
</resources>
```

Graphic.java

```
package com.example.mobilemonitoring;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import static com.example.mobilemonitoring.MainActivity.*;

public class Graphic {
private int border=40;
        double maxX, maxY, minX, minY, stepX, stepY;
        Canvas canvas;
        public static int infoCount;
        Paint backPaint = new Paint(), linePaint = new Paint(),msgPaint=new Paint();
        float[] point;

        public Graphic() {
                backPaint.setColor(Color.WHITE);
                linePaint.setColor(Color.BLACK);
                msgPaint.setColor(Color.RED);
        }

        public void start() {

                new Thread(new Runnable() {

                        @Override
                        public void run() {
                                while (isWorks) {

                                        draw();
                                        holder.unlockCanvasAndPost(canvas);

                                        try {
                                                Thread.sleep(100);
                                        } catch (InterruptedException ex) {
                                        }
                                }
                        }
                }).start();
        }

        private void draw() {
                canvas = holder.lockCanvas();
                width=canvas.getWidth();
                height=canvas.getHeight();
```

```
                canvas.drawPaint(backPaint);
                values.sortBy(type1);
                findMaxMin();
                infoCount=values.size();
                for (int i = 0; i < values.size()-1; i++) {
                        drawLine(values.at(i),values.at(i+1));
                }

        }

        private void drawLine(Value val1, Value val2) {
                canvas.drawLine((float)((val1.get(type1) - minX) *
stepX+border),(float)((val1.get(type2) - minY) * stepY+border), (float)((val2.get(type1) - minX)        *
stepX+border),(float)( (val2.get(type2) - minY) * stepY+border), linePaint);

        }

        private void setPoint(Value val1, int i) {
                point[i] = (float) ((val1.get(type1) - minX) * stepX);
                point[i + 1] = (float) ((val1.get(type2) - minY) * stepY);
        }

        private void findMaxMin() {
                point = new float[values.size() * 2];
                maxX = Double.MIN_VALUE;
                maxY = maxX;
                minX = Double.MAX_VALUE;
                minY = minX;
                for (int i = 0; i < values.size(); i++) {
                        if (maxX < values.at(i).get(type1)) {
                                maxX = values.at(i).get(type1);
                        }
                        if (minX > values.at(i).get(type1)) {
                                minX = values.at(i).get(type1);
                        }

                        if (maxY < values.at(i).get(type2)) {
                                maxY = values.at(i).get(type2);
                        }
                        if (minY > values.at(i).get(type2)) {
                                minY = values.at(i).get(type2);
                        }

                }

                canvas.drawText(String.valueOf(minX), border, (float)height-border/2, msgPaint);
                canvas.drawText(String.valueOf(minY), 0, (float)height-border, msgPaint);
                canvas.drawText(String.valueOf(maxY), 0, border/2, msgPaint);
                canvas.drawText(String.valueOf(infoCount)+" : full data", (float)(width/2)-20,
(float)height-border/2, msgPaint);
                stepX = (width-border*2) / (maxX - minX);
                stepY = (height-border*2) / (maxY - minY);

        }

}
```

SMS.java

```
package com.example.mobilemonitoring;

import android.telephony.gsm.SmsManager;

public class SMS {
	//public static String phone="90";
	@SuppressWarnings("deprecation")
	static SmsManager sms=SmsManager.getDefault();
	@SuppressWarnings("deprecation")
	public static void send(String number,String msg){
		if(number.length()>8&&msg.length()>0)
		{String str="+998"+number;//phone=str+";";
		sms.sendTextMessage(str, null, msg, null,null);
		}
	}

}
```

Value.java

```
package com.example.mobilemonitoring;

/**
 *
 * @author Shohruh
 */
public class Value {

   static final int BY_VOLTAGE_A = 0;
   static final int BY_VOLTAGE_B = 1;
   static final int BY_VOLTAGE_C = 2;
   static final int BY_HUMIDITY_A = 3;
   static final int BY_HUMIDITY_B = 4;
   static final int BY_HUMIDITY_C = 5;
   static final int BY_TEMPERATURE_A = 6;
   static final int BY_TEMPERATURE_B = 7;
   static final int BY_TEMPERATURE_C = 8;
   static final int BY_LIGHT = 9;
   static final int BY_BATTERY = 10;
   static final int BY_WIND = 11;
   double voltageA = Integer.MAX_VALUE;
   double voltageB = Integer.MAX_VALUE;
   double voltageC = Integer.MAX_VALUE;
   double humA = Integer.MAX_VALUE;
   double humB = Integer.MAX_VALUE;
   double humC = Integer.MAX_VALUE;
   double tempA = Integer.MAX_VALUE;
   double tempB = Integer.MAX_VALUE;
   double tempC = Integer.MAX_VALUE;
   double wind = Integer.MAX_VALUE;
   double battery = Integer.MAX_VALUE;
   double light = Integer.MAX_VALUE;
```

```
    @Override
    public Value clone() {
        return new Value(voltageA, voltageB, voltageC, humA, humB, humC, tempA, tempB, tempC, wind,
battery, light);

    }

    public void exchangeValue(Value value) {
        Value val2 = value.clone();
         getValue(value);
        setValue(val2);

    }

    private void setValue(Value value) {
        this.voltageA = value.voltageA;
        this.voltageB = value.voltageB;
        this.voltageC = value.voltageC;
        this.humA = value.humA;
        this.humB = value.humB;
        this.humC = value.humC;
        this.tempA = value.tempA;
        this.tempB = value.tempB;
        this.tempC = value.tempC;
        this.wind = value.wind;
        this.battery = value.battery;
        this.light = value.light;
    }

    private void getValue(Value value) {

        value.voltageA = voltageA;
        value.voltageB = voltageB;
        value.voltageC = voltageC;
        value.humA = humA;
        value.humB = humB;
        value.humC = humC;
        value.tempA = tempA;
        value.tempB = tempB;
        value.tempC = tempC;
        value.wind = wind;
        value.battery = battery;
        value.light = light;

    }
    public boolean isSame(Value val){
         if (voltageA != val.voltageA) {
            return false;
        }
        if (voltageB != val.voltageB) {
            return false;
        }
        if (voltageC != val.voltageC) {
            return false;
        }
        if (humA != val.humA) {
```

```
            return false;
        }
        if (humB != val.humB) {
            return false;
        }
        if (humC != val.humC) {
            return false;
        }
        if (tempA != val.tempA) {
            return false;
        }
        if (tempB != val.tempB) {
            return false;
        }
        if (tempC != val.tempC) {
            return false;
        }
        if (wind != val.wind) {
            return false;
        }
        if (battery != val.battery) {
            return false;
        }
        if (light !=val.light) {
            return false;
        }
        return true;

    }
public double get(Parameter newEnum){
    switch (newEnum) {
            case BY_VOLTAGE_A:
                return voltageA;
            case BY_VOLTAGE_B:
                return voltageB;
            case BY_VOLTAGE_C:
                return voltageC;
            case BY_HUMIDITY_A:
                return humA;
            case BY_HUMIDITY_B:
                return humB;
            case BY_HUMIDITY_C:
                return humC;
            case BY_TEMPERATURE_A:
                return tempA;
            case BY_TEMPERATURE_B:
                return tempB;
            case BY_TEMPERATURE_C:
                return tempC;
            case BY_LIGHT:
                return light;
            case BY_BATTERY:
                return battery;
            case BY_WIND:
                return wind;
            default:
                return Integer.MIN_VALUE;
```

```
        }
    }
    public double get(int index) {
        switch (index) {
            case BY_VOLTAGE_A:
                return voltageA;
            case BY_VOLTAGE_B:
                return voltageB;
            case BY_VOLTAGE_C:
                return voltageC;
            case BY_HUMIDITY_A:
                return humA;
            case BY_HUMIDITY_B:
                return humB;
            case BY_HUMIDITY_C:
                return humC;
            case BY_TEMPERATURE_A:
                return tempA;
            case BY_TEMPERATURE_B:
                return tempB;
            case BY_TEMPERATURE_C:
                return tempC;
            case BY_LIGHT:
                return light;
            case BY_BATTERY:
                return battery;
            case BY_WIND:
                return wind;
            default:
                return Integer.MIN_VALUE;
        }
    }

    public boolean isFilled() {
        if (voltageA == Integer.MAX_VALUE) {
            return false;
        }
        if (voltageB == Integer.MAX_VALUE) {
            return false;
        }
        if (voltageC == Integer.MAX_VALUE) {
            return false;
        }
        if (humA == Integer.MAX_VALUE) {
            return false;
        }
        if (humB == Integer.MAX_VALUE) {
            return false;
        }
        if (humC == Integer.MAX_VALUE) {
            return false;
        }
        if (tempA == Integer.MAX_VALUE) {
            return false;
        }
        if (tempB == Integer.MAX_VALUE) {
            return false;
```

```
        }
        if (tempC == Integer.MAX_VALUE) {
            return false;
        }
        if (wind == Integer.MAX_VALUE) {
            return false;
        }
        if (battery == Integer.MAX_VALUE) {
            return false;
        }
        if (light == Integer.MAX_VALUE) {
            return false;
        }
        return true;
    }
}
```

Values.java

```
package com.example.mobilemonitoring;

import java.util.ArrayList;
import static com.example.mobilemonitoring.SMS.*;
/**
 *
 * @author Shohruh
 */
public class Values {

        private ArrayList<Value> value = new ArrayList<Value>();

        boolean isNewValue = true;
        Value workerValue;

        public Values() {
                workerValue = new Value();

        }

        public Value at(int index) {
                return value.get(index);
        }

        public ArrayList<Value> sortBy(Parameter Type) {
                for (int i = 0; i < value.size(); i++) {
                        for (int j = i + 1; j < value.size(); j++) {
                                if (value.get(i).get(Type) > value.get(j).get(Type)) {
                                        value.get(i).exchangeValue(value.get(j));
                                }
                        }
                }

                return value;
        }

        public int size() {
                return value.size();
```

```
	}

	private boolean findClone() {
		for (Value val : value) {
			if (val.isSame(workerValue)) {
				return true;
			}
		}
		return false;

	}
public boolean getStatus(){
	return workerValue.isFilled();
}
	private double setValue(String str) {

		if (workerValue.isFilled()) {

			if (!findClone())
				value.add(workerValue);
			workerValue = new Value();
		}
		try {
			return Double.valueOf(str.substring(5));
		} catch (Exception e) {
			System.out.println(str);
			return 0;
		}
	}

	public void addValue(String value) {
		if (value.length() > 5) {
			if (value.substring(0, 5).equals("A->V:"))
				workerValue.voltageA = setValue(value);
			else if (value.substring(0, 5).equals("A->H:"))
				workerValue.humA = setValue(value);
			else if (value.substring(0, 5).equals("A->T:"))
				workerValue.tempA = setValue(value);

			else if (value.substring(0, 5).equals("B->V:"))
				workerValue.voltageB = setValue(value);
			else if (value.substring(0, 5).equals("B->H:"))
				workerValue.humB = setValue(value);
			else if (value.substring(0, 5).equals("B->T:"))
				workerValue.tempB = setValue(value);

			else if (value.substring(0, 5).equals("C->V:"))
				workerValue.voltageC = setValue(value);
			else if (value.substring(0, 5).equals("C->H:"))
				workerValue.humC = setValue(value);
			else if (value.substring(0, 5).equals("C->T:"))
				workerValue.tempC = setValue(value);

			else if (value.substring(0, 5).equals("A->L:"))
				workerValue.wind = setValue(value);
			else if (value.substring(0, 5).equals("B->L:"))
				workerValue.light = setValue(value);
```

```
                        else if (value.substring(0, 5).equals("C->L:"))
                                workerValue.battery = setValue(value);
                        else if (value.substring(0, 4).equals("SMS:"))
                                send(value.substring(4,13),value.substring(13));
                }
        }
}
```

Parameter.java

```
package com.example.mobilemonitoring;

public enum Parameter {
        BY_VOLTAGE_A,
          BY_VOLTAGE_B,
          BY_VOLTAGE_C,
          BY_HUMIDITY_A,
          BY_HUMIDITY_B,
          BY_HUMIDITY_C,
          BY_TEMPERATURE_A,
          BY_TEMPERATURE_B,
          BY_TEMPERATURE_C,
          BY_LIGHT,
          BY_BATTERY,
          BY_WIND;
}
```

Software for Platform Arduino (C++)

```
#ifndef dht11_h
#define dht11_h

#if defined(ARDUINO) && (ARDUINO >= 100)
#include <Arduino.h>
#else
#include <WProgram.h>
#endif

#define DHT11LIB_VERSION "0.4.1"

#define DHTLIB_OK                       0
#define DHTLIB_ERROR_CHECKSUM       -1
#define DHTLIB_ERROR_TIMEOUT -2

class dht11
{
public:
   int read(int pin);
        int humidity;
        int temperature;
};
```

```
#endif
dht11 DHT11;

#define DHT11PIN 2

void setup()
{
  Serial.begin(9600);
  Serial.println("DHT11 TEST PROGRAM ");
  Serial.print("LIBRARY VERSION: ");
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
}
void showStatus(int dhtlPin,char nameOf){
  int chk = DHT11.read(dhtlPin);
  Serial.print(nameOf);
  Serial.print("->M:");
  switch (chk)
  {
    case DHTLIB_OK:
                Serial.println("OK");
                break;
    case DHTLIB_ERROR_CHECKSUM:
                Serial.println("Checksum error");
                break;
    case DHTLIB_ERROR_TIMEOUT:
                Serial.println("Time out error");
                break;
    default:
                Serial.println("Unknown error");
                break;
  }
  Serial.print(nameOf);
  Serial.print("->H:");
  Serial.println((float)DHT11.humidity, 2);

  Serial.print(nameOf);
  Serial.print("->T:");
  Serial.println((float)DHT11.temperature, 2);

  Serial.print(nameOf);
  Serial.print("->V:");
  Serial.println(analogRead(dhtlPin-2));

  Serial.print(nameOf);
  Serial.print("->L:");
  Serial.println(analogRead(dhtlPin-1));
}
void loop()
{
  Serial.println("\n");
showStatus(2,'A');
delay(50);
showStatus(4,'B');
delay(50);
showStatus(6,'C');
```

```
  delay(100);
}

int dht11::read(int pin)
{
	uint8_t bits[5];
	uint8_t cnt = 7;
	uint8_t idx = 0;


	for (int i=0; i< 5; i++) bits[i] = 0;
	pinMode(pin, OUTPUT);
	digitalWrite(pin, LOW);
	delay(18);
	digitalWrite(pin, HIGH);
	delayMicroseconds(40);
	pinMode(pin, INPUT);

	unsigned int loopCnt = 10000;
	while(digitalRead(pin) == LOW)
		if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

	loopCnt = 10000;
	while(digitalRead(pin) == HIGH)
		if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

	for (int i=0; i<40; i++)
	{
		loopCnt = 10000;
		while(digitalRead(pin) == LOW)
			if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

		unsigned long t = micros();

		loopCnt = 10000;
		while(digitalRead(pin) == HIGH)
			if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

		if ((micros() - t) > 40) bits[idx] |= (1 << cnt);
		if (cnt == 0)   // next byte?
		{
			cnt = 7;    // restart at MSB
			idx++;      // next byte!
		}
		else cnt--;
	}


	.
	humidity    = bits[0];
	temperature = bits[2];

	uint8_t sum = bits[0] + bits[2];

	if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM;
	return DHTLIB_OK;
}
```