MINISTRY FOR DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATION OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY INFORMATION TECHNOLOGIES

Admit to protection
Head of department
Amirsaidov U.B.

_____

2015 y._____

FINAL QUALIFICATION WORK OF BACHELOR DEGREE

On the theme:

"EMBEDDED COMMUNICATION SYSTEM DEVELOPMENT ON THE ARDUINO UNO PLATFORM FOR DIGITAL INTERFACE"

Graduating student_____ S. F. Fakhritdinov
(signature)

Teacher _____ M.M. Abdullayev
(signature)

Reviewer_____ U.X. Alayev
(signature)

Adviser on LSE_____ F. M. Qodirov
(signature)

Tashkent 2015

MINISTRY FOR DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATION OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

Faculty "Telecommunications technologies" Department "Data Transmission Network and Systems" Specialization 53113000 – Telecommunications

APPROVED

by Head of Department

«___» _____2015 y.

T A S K

Of final qualification work of student

Ruziqulov Asliddin Fazliddin ugli

Theme of the work: "Designing Device Control System Through Available Open Source Hardware Core"

1. Confirmed by order of the University from «25»12.2014 y. № 1467

2. Term of delivery of the finished work 13.06.2015

3. Initial data of work: Books, internet resource, journals, publications

4. The maintenance of a settlement – explanatory note (the list of questions subject to working out) 1. Embedded systems: usage and development;

2. Arduino based embedded systems and programming tools;

3. Designing embedded communication system by using digital interface of arduino uno platform;

4. Safety of vital activity and ecology.

5. The list of a graphic material: Demonstration slide, screenshot, images.

6. Data of task delivery 10.01.2015

Teacher_____

(signature)

Task accepted_____

(signature)

7. Adviser for separate sections of final work

Name of the chapter	Adviser	Signature, data	
		Given mission	Received mission
1. The main part of research **2.** Safety of vital activity and ecology	O.I. Urmonov F. M. Qodirov	10.01.2015 y. 19.05.2015 y.	10.01.2015 y. 19.05.2015 y.

8. Progress chart

№	Name of the chapter	Due date	Signature of teacher and adviser
1.	Embedded systems: usage and development	30.01.2015 y.	
2.	Arduino based embedded systems and programming tools	25.02.2015 y.	
3.	Designing embedded communication system by using digital interface of arduino uno platform	04.04.2015 y.	
4.	Safety of vital activity and ecology	02.06.2015 y.	

Graduating student _____ S. F. Fakhriddinov «__»_____2015 y
(signature)

Teacher _____ M.M. Abdullayev «__»_____2015 y
(signature)

The Final Qualification Work presented to your attention is dedicated to building and construction principles of embedded systems and their parts in communication. The research organized and material related to this topic gathered into three specific chapters, namely embedded system usage and development, Arduino based embedded systems and programming tools and last but not least, designing embedded communication system by using digital interface of Arduino Uno platform.

Apart from abovementioned information, in the fourth chapter there can be seen many precautions and suggestion to prevent any life safety and ecology accidents.

Данная ВКР посвящена исследованию встроенных систем, проведена их классификация, а также рассмотрены области применения встроенных систем. Подробно рассмотрены возможности программирования встроенных систем, построенных на базе Arduino. На базе платформы Arduino Uno спроектирована встроенная система и программное обеспечение для управления доступом системы связи на базе цифрового интерфейса.

Также в ВКР рассмотрены вопросы обеспечения безопасности и экологии систем связи.

Эътиборингизга тақдим этилаётган малакавий битирув иши ўррнатилган тизимлар ва уларнинг замонавий алова тизимларидаги ўрни ҳақида тахлиллар келтириб ўтилган. Тадқиқот жараёни ва ушбу мавзуга оид маълумотлар асосан учта бўлимга тақсимланган бўлиб, биринчи бўлимда ўрнатилган тизимларнинг ривожи, иккинчи бўлимда эса Arduino асосида ишловчи ўрнатилган тизимлар ва сўнги лекин асосий учинчи бўлимда эса Arduino Uno платформасида яратилган ўрнатилган алоқа тизимларининг рақамли интерфейсини шакллантириш кўриб чиқилган.

Юқоридагиларга қўшимча сифатида, саноат ва хусусан алоқа тизимларида хаёт фаолияти хавфсизлиги ва экология масалалари ҳам тўртинчи бўлимда келтириб ўтилган.

CONTENT

INTRODUCTION

Embedded systems are omnipresent and play significant roles in modern-day life. Embedded systems are alsodiverse and can be found in consumer electronics, such as digital cameras, DVD players and printers; in industrial robots; in advanced avionics, such as missile guidance systems and flight control systems; in medical equipment, suchas cardiac arrhythmia monitors and cardiac pacemakers; in automotive designs, such as fuel injection systems andauto-braking systems. Embedded systems have significantly improved the way we live today-and will continue tochange the way we live tomorrow.

Programming embedded systems is a special discipline, and demands that embedded systems developers have working knowledge of a multitude of technology areas. These areas range from low-level hardware devices, compiler technology, and debugging techniques, to the inner workings of real-time operating systems and multithreaded application design. These requirements can be overwhelming to programmers new to the embedded world. The learning process can be long and stressful. As such, I felt compelled to share my knowledge and experiences through practical discussions and illustrations in jumpstarting your embedded systems projects.

Some developers use a more traditional approach, focusing solely on programming low-level drivers and software thatcontrol the underlying hardware devices. Other books provide a high-level abstract approach using object-oriented methodologies and modeling languages. This diploma work however, concentrates on bridging the gap between thehigher-level abstract modeling concepts and the lower-level fundamental programming aspects of embedded systems development. The discussions carried throughout this work are based on years of experience gained from design and implementation of embedded systems, lessons learnt from previous mistakes, wisdom passed down fromothers, and results obtained from academic research. These elements join together to form useful insights, guidelines

and recommendations that you can actually use in your real-time embedded systems projects.

1. EMBEDDED SYSTEMS: USAGE AND DEVELOPMENT

1.1 Embedded system concepts and real examples

In ways virtually unimaginable just a few decades ago, embedded systems are reshaping the way people live, work and play. Embedded systems come in an endless variety of types, each exhibiting unique characteristics. For example, most vehicles driven today embed intelligent computer chips that perform value-added tasks, which makethe vehicles easier, cleaner, and more fun to drive. Telephone systems rely on multiple integrated hardware andsoftware systems to connect people around the world. Even private homes are being filled with intelligent appliances and integrated systems built around embedded systems, which facilitate and enhance everyday life.

Often referred to as pervasive or ubiquitous computers, embedded systems represent a class of dedicated computer systems designed for specific purposes. Many of these embedded systems are reliable and predictable.The devices that embed them are convenient, user-friendly, and dependable.

One special class of embedded systems is distinguished from the rest by its requirement to respond to external eventsin real time. This category is classified as the real-time embedded system.

As an introduction to embedded systems and real-time embedded systems, this chapter focuses on:

- examples of embedded systems,
- defining embedded systems,
- defining embedded systems with real-time behavior, and
- current trends in embedded systems.

Even though often nearly invisible, embedded systems are ubiquitous. Embedded systems are present in many industries, including industrial automation,

defense, transportation, and aerospace. For example, NASA's Mars Path Finder, Lock heed Martin's missile guidance system, and the Ford automobile all contain numerous embedded systems.

Every day, people throughout the world use embedded systems without even knowing it. In fact, the embedded system s invisibility is its very beauty: users reap the advantages without having to understand the intricacies of the technology [2].

Remarkably adaptable and versatile, embedded systems can be found at home, at work, and even in recreational devices. Indeed, it is difficult to find a segment of daily life that does not involve embedded systems in some way.Some of the more visible examples of embedded systems are provided in the next sections.

1.1.1 Defining the Embedded System

Some texts define embedded systems as computing systems or devices without a keyboard, display, or mouse.These texts use the look characteristic as the differentiating factor by saying, embedded systems do not look likeordinary personal computers; they look like digital cameras or smart toasters. These statements are all misleading.

A general definition of embedded systems is: embedded systems are computing systems with tightly coupled hardware and software integration that are designed to perform a dedicated function. The word embedded reflects the fact that these systems are usually an integral part of a larger system, known as the embedding system. Multiple embedded systems can coexist in an embedding system.

This definition is good but subjective. In the majority of cases, embedded systems are truly embedded, i.e., they are systems within systems. They either cannot or do not function on their own. Take, for example, the digital set-topbox (DST) found in many home entertainment systems nowadays. The digital audio/video decoding system, calledthe A/V decoder, which is an integral part of the DST, is an embedded system. The A/V decoder accepts a single multimedia

stream and produces sound and video frames as output. The signals received from the satellite by theDST contain multiple streams or channels. Therefore, the A/V decoder works in conjunction with the transportstream decoder, which is yet another embedded system. The transport stream decoder de-multiplexes the incoming multimedia streams into separate channels and feeds only the selected channel to the A/V decoder [2].

In some cases, embedded systems can function as standalone systems. The network router illustrated in Figure 1.2isa standalone embedded system. It is built using a specialized communication processor, memory, a number ofnetwork access interfaces (known as network ports), and special software that implements packet routing algorithms.In other words, the network router is a standalone embedded system that routes packets coming from one port toanother, based on a programmed routing algorithm.

The definition also does not necessarily provide answers to some often-asked questions. For example: Can apersonal computer be classified as an embedded system? Why? Can an Apple iBook that is used only as a DVDplayer be called an embedded system?

A single comprehensive definition does not exist. Therefore, we need to focus on the characteristics of embedded systems from many different perspectives to gain a real under-standing of what embedded systems are and whatmakes embedded systems special.

1.1.2 Embedded Systems in the Home Environment

Hidden conveniently within numerous household appliances, embedded systems are found all over the house. Consumers enjoy the effort-saving advanced features and benefits provided by these embedded technologies.

As shown in Figure 1.1, embedded systems in the home assume many forms, including security systems, cable andsatellite boxes for televisions, home theater systems, and telephone answering machines. As advances in microprocessors

continue to improve the functionality of ordinary products, embedded systems are helping drive the development of additional home-based innovations.

Figure 1.1. Embedded systems at home

1.1.3 Embedded Systems in the Work Environment

Embedded systems have also changed the way people conduct business. Perhaps the most significant example is theInternet, which is really just a very large collection of embedded systems that are interconnected using various networking technologies. Figure 1.2illustrates what a small segment of the Internet might look like.

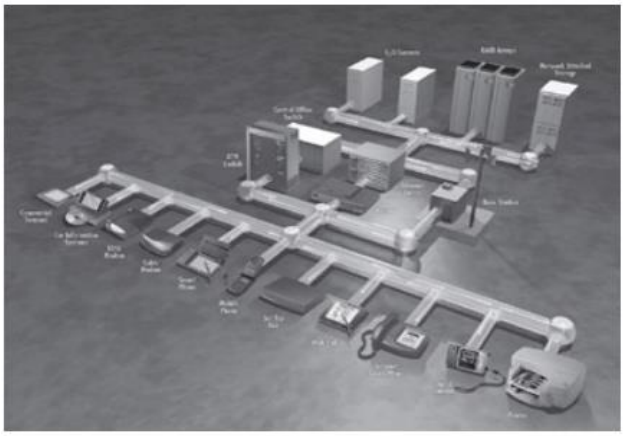

Figure 1.2. Embedded systems at work.

From various individual network end-points (for example, printers, cable modems, and enterprise network routers)to the backbone gigabit switches, embedded technology has helped make use of the Internet necessary to anybusiness model. The network routers and the backbone gigabit switches are examples of real-time embeddedsystems. Advancements in real-time embedded technology are making Internet connectivity both reliable andresponsive, despite the enormous amount of voice and data traffic carried over the network [3].

1.1.4 Embedded Systems in Leisure Activities

At home, at work, even at play, embedded systems are flourishing. A child s toy unexpectedly springs to life with unabashed liveliness. Automobiles equipped with in-car navigation systems transport people to destinations safelyand efficiently. Listening to favorite tunes with anytime-anywhere freedom is readily achievable, thanks to embedded systems buried deep within sophisticated portable music players, as shown in Figure 1.3.

Figure 1.3. Navigation system and portable music player

Even the portable computing device, called a web tablet, shown in Figure 1.4, is an embedded system.

Figure 1.4. A web tablet

Embedded systems also have teamed with other technologies to deliver benefits to the traditionally low-tech world.GPS technology, for example, uses satellites to pinpoint locations to centimeter-level accuracy, which allows hikers, cyclists, and other outdoor enthusiasts to use GPS handheld devices to enjoy vast spaces without getting lost. Even fishermen use GPS devices to store the locations of their favorite fishing holes.

Embedded systems also have taken traditional radio-controlled airplanes, racecars, and boats to new heights andspeeds. As complex embedded systems in disguise, these devices take command inputs from joysticks and passthem wirelessly to the device s receiver, enabling the model airplane, racecar, or boat to engage in speedy andcomplex maneuvers. In fact, the introduction of embedded technology has rendered these sports safer and more enjoyable for model owners by virtually eliminating the once-common threat of crashing due to signal interference.

1.1.5 Embedded Processor and Application Awareness

The processors found in common personal computers (PC) are general-purpose or universal processors. They arecomplex in design because these processors provide a full scale of features and a wide spectrum of functionalities. They are designed to be suitable for a variety of applications. The systems using these universal processors are programmed with a multitude of applications. For example, modern processors have a built-in memory management unit (MMU) to provide memory protection and virtual memory for multitasking-capable, general-purpose operatingsystems. These universal processors have advanced cache logic. Many of these processors have a built-in mathco-processor capable of performing fast floating-point operations. These processors provide interfaces to support avariety of external peripheral devices. These processors result in large power consumption, heat production, and size.The complexity means these processors are also expensive to fabricate. In the early days, embedded systems werecommonly built using general-purpose processors.

Because of the quantum leap in advancements made in microprocessor technology in recent years, embeddedsystems are increasingly being built using embedded processors instead of general-purpose processors. Theseembedded processors are special-purpose processors designed for a specific class of applications. The key isapplication awareness, i.e., knowing the nature of the applications and meeting the requirement for those applicationsthat it is designed to run.

One class of embedded processors focuses on size, power consumption, and price. Therefore, some embeddedprocessors are limited in functionality, i.e., a processor is good enough for the class of applications for which it wasdesigned but is likely inadequate for other classes of applications. This is one reason why many embeddedprocessors do not have fast CPU speeds. For example, the processor chosen for a personal digital assistant (PDA) device does not have a floating-point co-processor because floating-point operations are either not needed orsoftware

emulation is sufficient. The processor might have a 16-bit addressing architecture instead of 32-bit, due toits limited memory storage capacity. It might have a 200MHz CPU speed because the majority of the applicationsare interactive and display-intensive, rather than computation-intensive. This class of embedded processors is smallbecause the overall PDA device is slim and fits in the palm of your hand. The limited functionality means reducedpower consumption and long-lasting battery life. The smaller size reduces the overall cost of processor fabrication.

On the other hand, another class of embedded processors focuses on performance. These embedded processors arepowerful and packed with advanced chip-design technologies, such as advanced pipeline and parallel processingarchitecture. These processors are designed to satisfy those applications with intensive computing requirements notachievable with general-purpose processors. An emerging class of highly specialized and high-performance embedded processors includes network processors developed for the network equipment and telecommunications industry. Overall, system and application speeds are the main concerns.

Yet another class of embedded processors focuses on all four requirements performance, size, power consumption, and price. Take, for example, the embedded digital signal processor (DSP) used in cell phones. Real-time voicecommunication involves digital signal processing and cannot tolerate delays. A DSP has specialized arithmetic units,optimized design in the memory, and addressing and bus architectures with multiprocessing capability that allow theDSP to perform complex calculations extremely fast in real time. A DSP outperforms a general-purpose processor running at the same clock speed many times over comes to digital signal processing. These reasons are why DSPs, instead of general-purpose processors, are chosen for cell phone designs. Even though DSPs are incredibly fast andpowerful embedded processors, they are reasonably priced, which keeps the overall prices of cell phones competitive. The battery from which

the DSP draws power lasts for hours and hours. A cell phone under $100 fits inhalf the palm-size of an average person at the time this book was written.

System-on-a-chip (SoC) processors are especially attractive for embedded systems. The SoC processor is comprised of a CPU core with built-in peripheral modules, such as a programmable general-purpose timer, programmable interrupt controller, DMA controller, and possibly Ethernet interfaces. Such a self-contained designallows these embedded processors to be used to build a variety of embedded applications without needing additional external peripheral devices, again reducing the overall cost and size of the final product.

Sometimes a gray area exists when using processor type to differentiate between embedded and non-embedded systems. It is worth noting that, in large-scale, high-performance embedded systems; the choice between embedded processors and universal microprocessors is a difficult one.

In high-end embedded systems, system performance in a predefined context outweighs power consumption and cost.The choice of a high-end, general purpose processor is as good as the choice of a high-end, specialized embedded processor in some designs. Therefore, using processor type alone to classify embedded systems may result in wrong classifications [4].

1.1.6 Hardware and Software Co-Design Model

Commonly both the hardware and the software for an embedded system are developed in parallel. Constant designfeedback between the two design teams should occur in this development model. The result is that each side can takeadvantage of what the other can do. The software component can take advantage of special hardware features togain performance. The hardware component can simplify module design if functionality can be achieved in softwarethat reduces overall hardware complexity and cost. Often design flaws, in both the hardware and software, areuncovered during this close collaboration.

The hardware and software co-design model reemphasizes the fundamental characteristic of embedded systems theyare application-specific. An embedded system is usually built on custom hardware and software. Therefore, using thisdevelopment model is both permissible and beneficial.

1.1.7 Cross-Platform Development

Another typical characteristic of embedded systems is its method of software development, called cross-platformdevelopment, for both system and application software. Software for an embedded system is developed on oneplatform but runs on another. In this context, the platform is the combination of hardware (such as particular type ofprocessor), operating system, and software development tools used for further development.

The host system is the system on which the embedded software is developed. The targetsystem is the embeddedsystem under development.

The main software tool that makes cross-platform development possible is a cross compiler. A cross compiler is acompiler that runs on one type of processor architecture but produces object code for a different type of processorarchitecture. A cross compiler is used because the target system cannot host its own compiler. For example, theDIAB compiler from Wind River Systems is such a cross compiler. The DIAB compiler runs on the MicrosoftWindows operating system (OS) on the IA-32 architecture and runs on various UNIX operating systems, such asthe Solaris OS on the SPARC architecture. The compiler can produce object code for numerous processor types, such as Motorola s 68000, MIPS, and ARM.

1.1.8 Software Storage and Upgradeability

Code for embedded systems (such as the real-time embedded operating system, the system software, and theapplication software) is commonly stored in ROM and NVRAM memory devices. We discuss theembedded system booting

process and the steps involved in extracting code from these storage devices. Upgradingan embedded system can mean building new PROM, deploying special equipment and/or a special method toreprogram the EPROM, or reprogramming the flash memory.

The choice of software storage device has an impact on development. The process to reprogram an EPROM whensmall changes are made in the software can be tedious and time-consuming, and this occurrence is common duringdevelopment. Removing an EPROM device from its socket can damage the EPROM; worse yet, the system itselfcan be damaged if careful handling is not exercised.

The choice of the storage device can also have an impact on the overall cost of maintenance. Although PROM andEPROM devices are inexpensive, the cost can add up if a large volume of shipped systems is in the field. Upgradingan embedded system in these cases means shipping replacement PROM and EPROM chips. The embedded systemcan be upgraded without the need for chip replacement and can be upgraded dynamically over a network if flashmemory or EEPROM is used as the code storage device (see the following sidebar).

Armed with the information presented in the previous sections, we can now attempt to answer the questions raisedearlier. A personal computer is not an embedded system because it is built using a general-purpose processor and isbuilt independently from the software that runs on it. The software applications developed for personal computers,which run operating systems such as FreeBSD or Windows, are developed natively (as opposed tocross-developed) on those operating systems. For the same reasons, an Apple iBook used only as a DVD player isused like an embedded system but is not an embedded system.

1.2 Real-Time Embedded Systems

In the simplest form, real-time systems can be defined as those systems that respond to external events in a timelyfashion, as shown in Figure 1.5. The response

time is guaranteed. We revisit this definition after presenting someexamples of real-time systems.

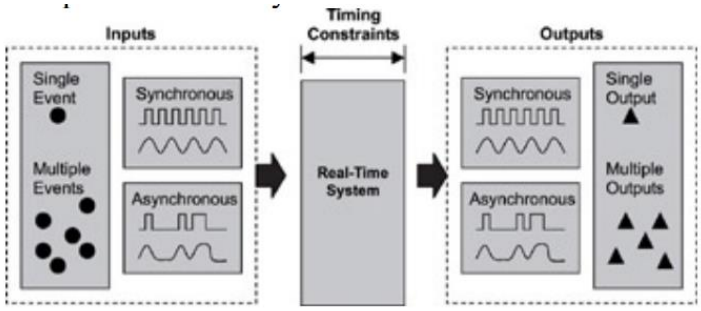

Figure 1.5. A simple view of real-time systems

External events can have synchronous or asynchronous characteristics. Responding to external events includesrecognizing when an event occurs, performing the required processing as a result of the event, and outputting thenecessary results within a given time constraint. Timing constraints include finish time, or both start time and finish time.

A good way to understand the relationship between real-time systems and embedded systems is to view them as twointersecting circles, as shown in Figure 1.6. It can be seen that not all embedded systems exhibit real-time behaviorsnor are all real-time systems embedded. However, the two systems are not mutually exclusive, and the area in whichthey overlap creates the combination of systems known as real-time embedded systems.

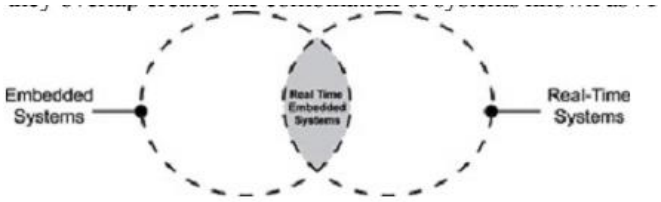

Figure 1.6. Real-time embedded systems

Knowing this fact and because we have covered the various aspects of embedded systems in the previous sections, we can now focus our attention on real-time systems.

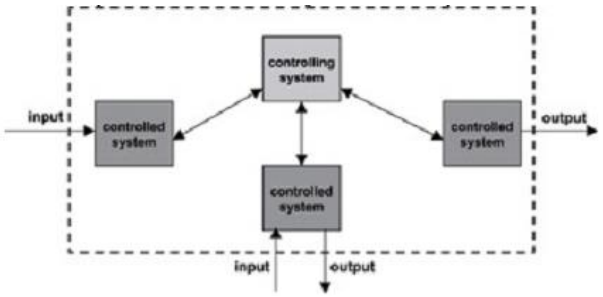

Figure 1.7. Structure of real-time systems

1.2.1 Real-Time Systems

The environment of the real-time system creates the external events. These events are received by one or morecomponents of the real-time system. The response of the real-time system is then injected into its environmentthrough one or more of its components. Decomposition of the real-time system, as shown in Figure 1.5, leads to thegeneral structure of real-time systems.

The structure of a real-time system, as shown in Figure 1.7, is a controlling system and at least one controlled system.The controlling system interacts with the controlled system in various ways. First, the interaction can be periodic, inwhich communication is initiated from the controlling system to the controlled system. In this case, the communicationis predictable and occurs at predefined intervals. Second, the interaction can be aperiodic, in which communicationis initiated from the controlled system to the controlling system. In this case, the communication is unpredictable and isdetermined by the random occurrences of external events in the environment of the controlled system. Finally, thecommunication can be a combination of both types. The controlling system must process and respond to the eventsand information generated by the controlled system in a guaranteed time frame.

Imagine a real-time weapons defense system whose role is to protect a naval destroyer by shooting down incomingmissiles. The idea is to shred an incoming missile into pieces with bullets before it reaches the ship. The weaponssystem is comprised of a radar system, a command-and-decision (C&D) system, and weapons firing control system.The controlling system is the C&D system, whereas the controlled systems are the radar system and the weaponsfiring control system.

•The radar system scans and searches for potential targets. Coordinates of a potential target are sent to theC&D system periodically with high frequency after the target is acquired.

•The C&D system must first determine the threat level by threat classification and evaluation, based on thetarget information provided by the radar system. If a threat is imminent, the C&D system must, at a minimum,calculate the speed and flight path or trajectory, as well as estimate the impact location. Because a missiletends to drift off its flight path with the degree of drift dependent on the precision of its guidance system, theC&D system calculates an area (a box) around the flight path.

•The C&D system then activates the weapons firing control system closest to the anticipated impact locationand guides the weapons system to fire continuously within the moving area or box until the target isdestroyed. The weapons firing control system is comprised of large-caliber, multi-barrel, high-muzzlevelocity, high-power machine guns.

In this weapons defense system example, the communication between the radar system and the C&D system isaperiodic, because the occurrence of a potential target is unpredictable and the potential target can appear at anytime. The communication between the C&D system and the weapons firing control system is, however, periodicbecause the C&D system feeds the firing coordinates into the weapons control system periodically (with an extremelyhigh frequency). Initial firing coordinates are based on a pre-computed flight path but are updated in real-timeaccording to the actual location of the incoming missile.

Consider another example of a real-time system-the cruise missile guidance system. A cruise missile flies at subsonicspeed. It can travel at about 10 meters above water, 30 meters above flat ground, and 100 meters above mountainterrains. A modern cruise missile can hit a target within a 50-meter range. All these capabilities are due to thehigh-precision, real-time guidance system built into the nose of a cruise missile. In a simplified view, the guidancesystem is comprised of the radar system (both forward-looking and look-down radars), the navigation system, andthe divert-and-altitude-control system. The navigation system contains digital maps covering the missile flight path.The forward-looking radar scans and maps out the approaching terrains. This information is fed to the navigationsystem in real time. The navigation system must then recalculate flight coordinates to avoid terrain obstacles. The newcoordinates are immediately fed to the divert-and-altitude-control system to adjust the flight path. The look-downradar periodically scans the ground terrain along its flight path. The scanned data is compared with the estimatedsection of the pre-recorded maps. Corrective adjustments are made to the flight coordinates and sent to thedivert-and-altitude-control system if data comparison indicates that the missile has drifted off the intended flight path.

In this example, the controlling system is the navigation system. The controlled systems are the radar system and thedivert-and-altitude-control system. We can observe both periodic and aperiodic communications in this example.The communication between the radars and the navigation system is aperiodic. The communication between thenavigation system and the diver-and-altitude-control system is periodic.

Let us consider one more example of a real-time system-a DVD player. The DVD player must decode both thevideo and the audio streams from the disc simultaneously. While a movie is being played, the viewer can activate theon-screen display using a remote control. On-screen display is a user menu that allows the user to changeparameters, such as the audio output format and language options. The DVD player is the controlling system, and theremote control is the

controlled system. In this case, the remote control is viewed as a sensor because it feeds events, such as pause and language selection, into the DVD player.

1.2.2 Characteristics of Real-Time Systems

The C&D system in the weapons defense system must calculate the anticipated flight path of the incoming missilequickly and guide the firing system to shoot the missile down before it reaches the destroyer. Assume T1 is the timethe missile takes to reach the ship and is a function of the missile's distance and velocity. Assume T2 is the time theC&D system takes to activate the weapons firing control system and includes transmitting the firing coordinates plusthe firing delay. The difference between T1 and T2 is how long the computation may take. The missile would reachits intended target if the C&D system took too long in computing the flight path. The missile would still reach its targetif the computation produced by the C&D system was inaccurate. The navigation system in the cruise missile mustrespond to the changing terrain fast enough so that it can re-compute coordinates and guide the altitude controlsystem to a new flight path. The missile might collide with a mountain if the navigation system cannot compute newflight coordinates fast enough, or if the new coordinates do not steer the missile out of the collision course.

Therefore, we can extract two essential characteristics of real-time systems from the examples given earlier. Thesecharacteristics are that real-time systems must produce correct computational results, called logical or functionalcorrectness , and that these computations must conclude within a predefined period, called timing correctness.

Real-time systemsare defined as those systems in which the overall correctness of the system depends on both thefunctional correctness and the timing correctness. The timing correctness is at least as important as the functionalcorrectness.

It is important to note that we said the timing correctness is at least as important as the functional correctness. Insome real-time systems, functional correctness is sometimes sacrificed for timing correctness. We address this pointshortly after we introduce the classifications of real-time systems.

Similar to embedded systems, real-time systems also have substantial knowledge of the environment of the controlledsystem and the applications running on it. This reason is one why many real-time systems are said to be deterministic, because in those real-time systems, the response time to a detected event is bounded. The action (or actions) takenin response to an event is known a priori. A deterministic real-time system implies that each component of the systemmust have a deterministic behavior that contributes to the overall determinism of the system. As can be seen, adeterministic real-time system can be less adaptable to the changing environment. The lack of adaptability can resultin a less robust system. The levels of determinism and of robustness must be balanced. The method of balancingbetween the two is system-and application-specific.

1.2.3 Hard and Soft Real-Time Systems

In the previous section, we said computation must complete before reaching a given deadline. In other words, real-time systems have timing constraints and are deadline-driven. Real-time systems can be classified, therefore, aseither hard real-time systems or soft real-time systems.

What differentiates hard real-time systems and soft real-time systems are the degree of tolerance of missed deadlines, usefulness of computed results after missed deadlines, and severity of the penalty incurred for failing to meet deadlines.

For hard real-time systems, the level of tolerance for a missed deadline is extremely small or zero tolerance. Thecomputed results after the missed deadline are likely useless for many of these systems. The penalty incurred for amissed deadline is catastrophe. For soft real-time systems, however, the level of tolerance is non-zero. Thecomputed results after the missed deadline have a rate of

depreciation. The usefulness of the results does not reachzero immediately passing the deadline, as in the case of many hard real-time systems. The physical impact of a misseddeadline is non-catastrophic.

A hard real-time system is a real-time system that must meet its deadlines with a near-zero degree of flexibility. Thedeadlines must be met, or catastrophes occur. The cost of such catastrophe is extremely high and can involve humanlives. The computation results obtained after the deadline have either a zero-level of usefulness or have a high rate ofdepreciation as time moves further from the missed deadline before the system produces a response.

A soft real-time system is a real-time system that must meet its deadlines but with a degree of flexibility. Thedeadlines can contain varying levels of tolerance, average timing deadlines, and even statistical distribution ofresponse times with different degrees of acceptability. In a soft real-time system, a missed deadline does not result insystem failure, but costs can rise in proportion to the delay, depending on the application.

Penalty is an important aspect of hard real-time systems for several reasons.

•What is meant by 'must meet the deadline'?

•It means something catastrophic occurs if the deadline is not met. It is the penalty that sets the requirement.

•Missing the deadline means a system failure, and no recovery is possible other than a reset, so the deadlinemust be met. Is this a hard real-time system? That depends. If a system failure means the system must be reset but no cost is associated with the failure, the deadline is not a hard deadline, and the system is not a hard real-time system. On the other hand, if a costis associated, either in human lives or financial penalty such as a $50 million lawsuit, the deadline is a harddeadline, and it is a hard real-time system. It is the penalty that makes this determination.

•What defines the deadline for a hard real-time system?

•It is the penalty. For a hard real-time system, the deadline is a deterministic value, and, for a soft real-timesystem, the value can be estimation.

One thing worth noting is that the length of the deadline does not make a real-time system hard or soft, but it is therequirement for meeting it within that time.

The weapons defense and the missile guidance systems are hard real-time systems. Using the missile guidance systemfor an example, if the navigation system cannot compute the new coordinates in response to approaching mountainterrain before or at the deadline, not enough distance is left for the missile to change altitude. This system has zerotolerance for a missed deadline. The new coordinates obtained after the deadline are no longer useful because atsubsonic speed the distance is too short for the altitude control system to navigate the missile into the new flight pathin time. The penalty is a catastrophic event in which the missile collides with the mountain. Similarly, the weaponsdefense system is also a zero-tolerance system. The missed deadline results in the missile sinking the destroyer, andhuman lives potentially being lost. Again, the penalty incurred is catastrophic.

On the other hand, the DVD player is a soft real-time system. The DVD player decodes the video and the audiostreams while responding to user commands in real time. The user might send a series of commands to the DVDplayer rapidly causing the decoder to miss its deadline or deadlines. The result or penalty is momentary but visiblevideo distortion or audible audio distortion. The DVD player has a high level of tolerance because it continues tofunction. The decoded data obtained after the deadline is still useful.

Timing correctness is critical to most hard real-time systems. Therefore, hard real-time systems make every effortpossible in predicting if a pending deadline might be missed. Returning to the weapons defense system, let us discusshow a hard real-time system takes corrective actions when it anticipates a deadline might be missed. In the weaponsdefense system example, the C&D system calculates a firing box around the projected missile flight path. The missilemust be destroyed a certain distance away from the ship or the shrapnel can still cause damage. If the C&D systemanticipates a missed deadline (for example, if by the time the precise

firing coordinates are computed, the missilewould have flown past the safe zone), the C&D system must take corrective action immediately. The C&D systemenlarges the firing box and computes imprecise firing coordinates by methods of estimation instead of computing forprecise values. The C&D system then activates additional weapons firing systems to compensate for this imprecision.The result is that additional guns are brought online to cover the larger firing box. The idea is that it is better to wastebullets than sink a destroyer.

This example shows why sometimes functional correctness might be sacrificed for timing correctness for manyreal-time systems.

Because one or a few missed deadlines do not have a detrimental impact on the operations of soft real-time systems, a soft real-time system might not need to predict if a pending deadline might be missed. Instead, the soft real-timesystem can begin a recovery process after a missed deadline is detected.

For example, using the real-time DVD player, after a missed deadline is detected, the decoders in the DVD playeruse the computed results obtained after the deadline and use the data to make a decision on what future video framesand audio data must be discarded to re-synchronize the two streams. In other words, the decoders find ways tocatch up.

So far, we have focused on meeting the deadline or the finish time of some work or job, e.g., a computation. Attimes, meeting the start time of the job is just as important. The lack of required resources for the job, such as CPUor memory, can prevent a job from starting and can lead to missing the job completion deadline. Ultimately thisproblem becomes a resource-scheduling problem. The scheduling algorithms of a real-time system must schedulesystem resources so that jobs created in response to both periodic and aperiodic events can obtain the resources atthe appropriate time. This process affords each job the ability to meet its specific timing constraints.

1.3 The Future of Embedded Systems

Until the early 1990s, embedded systems were generally simple, autonomous devices with long product lifecycles. Inrecent years, however, the embedded industry has experienced dramatic transformation, as reported by the GartnerGroup, an independent research and advisory firm, as well as by other sources:

•Product market windows now dictate feverish six- to nine-month turnaround cycles.

•Globalization is redefining market opportunities and expanding application space.

•Connectivity is now a requirement rather than a bonus in both wired and emerging wireless technologies.

•Electronics-based products are more complex.

•Interconnecting embedded systems are yielding new applications that are dependent on networkinginfrastructures.

•The processing power of microprocessors is increasing at a rate predicted by Moore s Law, which statesthat the number of transistors per integrated circuit doubles every 18 months.

If past trends give any indication of the future, then as technology evolves, embedded software will continue toproliferate into new applications and lead to smarter classes of products. With an ever-expanding marketplacefortified by growing consumer demand for devices that can virtually run themselves as well as the seemingly limitlessopportunities created by the Internet, embedded systems will continue to reshape the world for years to come.

SUMMARY

An embedded system is built for a specific application. As such, the hardware and software components arehighly integrated, and the development model is the hardware and software co-design model.

•Embedded systems are generally built using embedded processors.

•An embedded processor is a specialized processor, such as a DSP, that is cheaper to design and produce,can have built-in integrated devices, is limited in functionality, produces low heat, consumes low power, anddoes not necessarily have the fastest clock speed but meets the requirements of the specific applications forwhich it is designed.

•Real-time systems are characterized by the fact that timing correctness is just as important as functional orlogical correctness.

•The severity of the penalty incurred for not satisfying timing constraints differentiates hard real-time systemsfrom soft real-time systems.

•Real-time systems have a significant amount of application awareness similar to embedded systems.

•Real-time embedded systems are those embedded system with real-time behaviors.

2.ARDUINO BASED EMBEDDED SYSTEMS AND PROGRAMMING TOOLS

2.1 Arduino based embedded hardware systems

You have probably heard the statement that everything is a computer nowadays. The use of microcontrollers responds to the paradigm of so-called ubiquitous computing. Computers are everywhere: the average car has 70 processors, your microwave has one, and your cellphone has a couple of them. You can find general-purpose processors that can be reprogrammed into making almost anything. You can also find purpose-specific processors, aimed at solving a certain type of issue such as USB-serial conversion, decoding MP3 sound files, or controlling the movement of a motor.Nowadays microcontrollers are just the intelligence of your embedded project. To gather data about the world, you need devices that translate real-world data into digital information. Sensors are the interfaces that translate properties of the world such as temperature, acceleration, or intensity of light. These are then translated into a voltage to be read by the microcontroller.

Arduino is just a microcontroller that can run software made in C. Even if it is possible, Arduino doesn't run an operating system. To keep the system simple, it runs sequential programs performing commands one by one. In contrast, Android is a whole operating system that can run on phones and tablets using many different vendors.

You can find many types of microcontrollers from several different vendors in the market. Don't get confused between the microcontroller platform and the microcontroller itself. Microcontrollers are the chips, the black boxes on the circuits. Platforms are the ecosystem integrated by the microcontroller — its circuit, the programming environment, and the documentation.

In that way, some platforms depend very much on the processor's architecture, whereas others can be implemented on technology from many different vendors. Arduino has been designed to run on different architectures. The API used to program the microcontrollers has been translated into processors coming from a whole series of vendors.

Arduino is a software abstraction that you can apply to much different architecture. You can move the same software seamlessly between boards in an upwards-compatible way. Some of the newest programs might contain features that cannot run on older boards. Different manufacturers (Atmel, Freescale, ST, Microchip, Texas Instruments) are making microcontrollers. Those chips have to beintegrated into platforms for people to use them. Those same vendors produce prototyping platforms for engineers to try out the chips and decide whether or not those processors could be used as part of a project.

Arduino is an external actor that can use any of those manufacturers to create prototyping boards. One of the main goals when learning about digital technology through Arduino is that it should be easy to use. That ease of use is achieved through a simplified IDE and curated documentation. Figure 2-1 shows the Arduino IDE running on a Mac computer. The IDE is cross-platform and can run on Windows, Linux and Mac computers. The code should compile the same way and the board should behave identically.

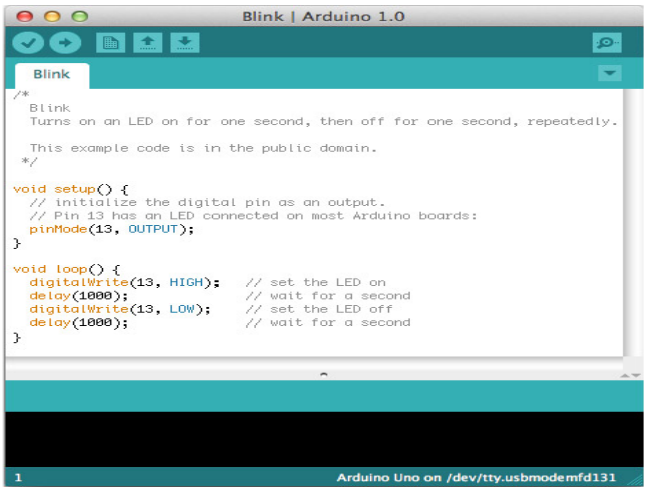

Figure 2.1. Arduino IDE

From an engineering point of view, you would analyze how to solve a project from the technical features in a brief. You would then choose the right microcontroller and IDE to program it.

Choosing a platform like Arduino enables you to start your project with one tool and, if that doesn't completely fulfill your needs in terms of available inputs/outputs, size, or speed, you could move into using a different platform without changing your code. It also enables you to go the other way around: You could start with your most powerful platform and once you are done, trim the project down to make it fit in as small a form factor as you need.

The following as shown in Table 2-1, give an overview of different prototyping boards and examples of using each one. Also, keep in mind that the Arduino boards are open-source hardware, which means that it should be possible to find platform vendors making Arduino-compatible boards with similar or equal functionality to the ones mentioned here.

Table 2.1. Comparing the Arduinoboards

Board	Architecture	Digital i/o	Analog i/o	ADK functionality	Serial ports
Arduino Uno	8-bits	14 pin	6 analog inputs, 6 digital pins do PWM	No	1
Arduino Mega ADK	8-bits	54 pins	16 analog inputs, 16 digital pins do PWM	Yes	4
Arduino Due	32-bits	54 pins	12 analog inputs, 12 digital pins do PWM,	Yes	4

			2 pins from DAC		
Arduino Micro ADK	8-bits	11 pins	6 analog inputs, 6 digital pins do PWM	Yes	1

2.1.1 Arduino Uno

This is the most extended board from the Arduino family. The board carries an 8-bit microcontroller, and it comes with 14 digital input/output pins and 6 analog inputs. Six of the digital pins can be programmed to send pulse width modulation (PWM). The Uno board also comes with internal peripherals able of running the UART, SPI, and I2C communication protocols. Programs using the Arduino Uno board (Figure 2-2) can be as big as 30 Kbytes and run at 16 MHz.

Figure 2.2. Arduino Uno board

As you can see, Arduino Uno is a very versatile board that has become the Swiss army knife of the maker community. In literally five minutes you can plug this board into your computer and start programming it. It has a disadvantage,

though: You cannot use it to communicate directly with an Android device, because it lacks the USB Host functionality. However, shields are available that can bring the USB Host functionality to the board, as well as some that could add Bluetooth communication. Both methods allow the microcontroller to talk to Android devices.

2.1.2 Arduino Mega ADK

The most extended of all the prototyping boards that can communicate with Android phones is probably the Arduino Mega ADK. This board, shown in Figure 2-3, includes all the functionality of an Arduino Mega 2560 board plus the USB Host. The Arduino Mega 2560, and derivatives, includes a chip with 252 KB of available memory space, 54 digital input/output pins (of which 16 can use PWM), 16 analog inputs, and 4 serial ports (UART). It is very convenient for projects that use sensors to communicate over serial (UART) or systems that require reading many inputs. Like its smaller brother (the Uno) also works at 16 MHz and uses an 8-bit processor.

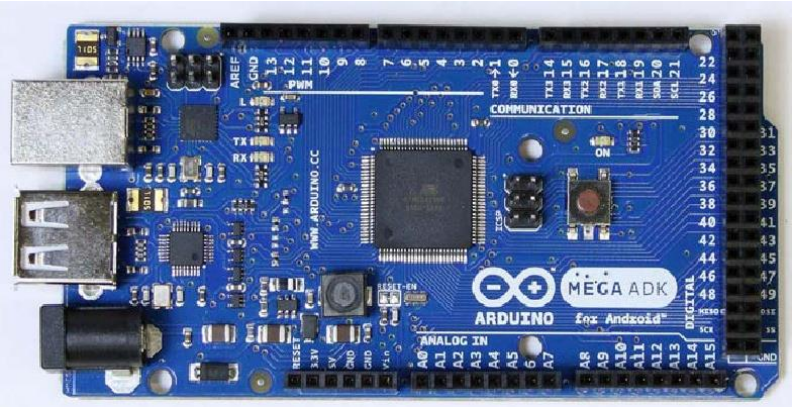

Figure 2.3 Arduino Mega ADK board

As mentioned, the Mega ADK adds to that the USB Host functionality. This means that you can connect a USB device through an on-board female USB connector. It is, of course, possible to connect other USB devices to the board such as Bluetooth dongles, keyboards, or USB drives.

2.1.3 Arduino Due

Arduino Due (Figure 2.4) is the first board within the Arduino family to leave the 8-bit realm. It runs an ARM core type Cortex-M3 with 32-bit, internal Digital Signal Processor (DSP), a 12-bit Digital to Analog Converter (DAC), UART, SPI, I2C, and other peripherals. Probably the most interesting of all its capabilities when it comes to the AOA is that the microcontroller has an internal USB OTG (On The Go) peripheral. This is what is used to connect to USB devices either as a host or as a client. In other words, the chip includes the possibility to connect to the Android phone directly.

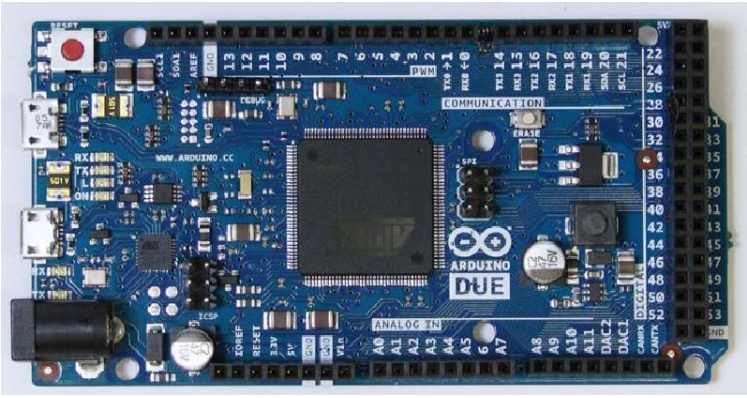

Figure 2.4. Arduino Due board

However, from a programming perspective, the experience is going to be the same as using the Arduino Mega ADK. The same code that runs in one of the

boards should run on the other, except for those cases when you might be using some of the extended features from the ARM core like DSP or internal DAC.

2.1.4 Arduino Micro ADK

Probably the only reason to not use any of the previously mentioned boards is their form factor. Both the Arduino Due and the Arduino Mega ADK have the same shape and size (5.33 3 10.16 cm or 2.1 3 4 inch). The Arduino Uno is shorter (6.85 cm or 2.7 inch), but in turn, you need to add the USB Host shield on top of it to achieve the desired functionality. Any of the combinations are optimal in terms of the prototyping experience. They have enough ports and pins to accommodate almost any project you could envision. But when you want to pack your project in a small form factor, you need to rethink how to put things together.

This is where the Arduino Micro ADK board (Figure 2.5) comes in. Its smaller form factor (2.3 Ч 6.85 cm or 0.9 Ч 2.7 inch) makes it perfect for many projects. It offers fewer pins and you need to power it up from a battery, but for wearable projects or those where you just want to control a couple of motors and hide everything in a small box, this might be the best tool for you. It also has some really nice features the other boards don’t have. Its programming port can transform the board into a computer keyboard or a mouse. You could imagine making an app that would type SMS messages in your computer, or execute certain actions using the command line.

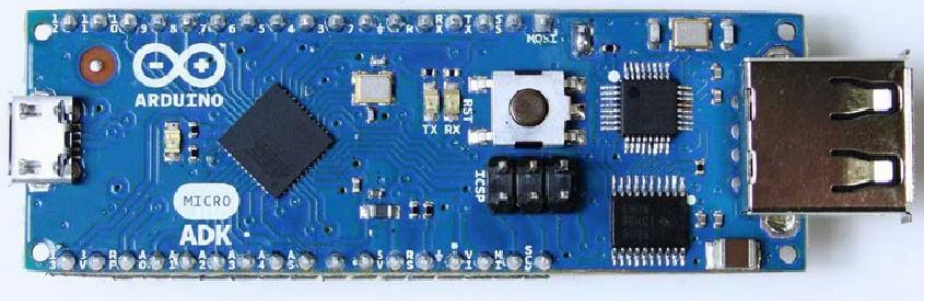

Figure 2.5 Arduino Micro ADK board

2.1.5 Arduino ADK vs. Google ADK Boards

Google's official ADK boards (the ADK from 2011 and the ADK2 from 2012) are proofs of concept presented by Google. The company from Mountain View is not interested in manufacturing and selling these platforms — it wants developers to get excited about making accessories for Android devices and therefore make reference designs available for others to take over and bring to the market.

The Google ADK (Figure 2.6) is a development kit presented by Google at its annual conference in May 2011. It is made of two parts:

➤ An Arduino Mega-compatible board that was made by adding a chip with USB Host functionality to the Arduino Mega reference design.

➤ A shield with a series of buttons, LEDs, a touch sensor shaped like the Android OS robot logotype, and a couple of relays to interact with the physical world.

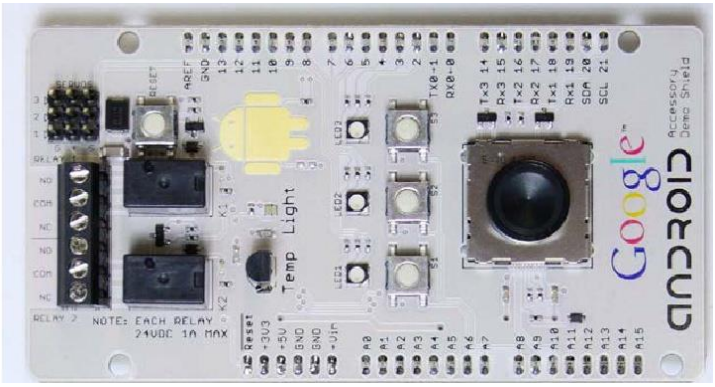

Figure 2.6 Google ADK Board

These boards were made in white with a multicolored silkprint showing Google's logotype. The whole kit came in a white cylindrical package. About

1,000 people listened to Google's live official presentation and left the building with one of these boards.

The development kit was made as a one-run production and was priced quite high. Arduino ADK is fully compatible with this system and can be easily obtained from more than 200 distributors around the world.

On the other hand, the Google ADK2 (Figure 2.7) is an evolution of the previous development kit, and was presented in San Francisco's Google IO of 2012. It is again made of two parts:

➤ An Arduino Due-compatible board with the extra feature of a Bluetooth series 4 chip, which enables you to make wireless accessories and a MicroSD card socket.

➤ A shield with a series of LEDs, inputs, and sound amplifier. It is shaped like a trapezoid.

Figure 2.7 Google ADK2 board

This time, Google's design team went for blue (for the microcontroller board) and dark blue (for the shield).

2.1.6 Shields of Arduino embedded board

Shields are boards put on top of the Arduino board. They enhance the platform's basic functionality by bringing in extra peripherals and sensors. Many shields come with libraries specifically made for them.

Many different shields are available. Arduino is an open platform, which allows makers and manufacturers to produce their own add-ons to the platform at no fee. You can find specific shields for almost any application out there: controlling motors, reading Real Time Clock (RTC) chips, storing data in SD cards, and so on. The following sections describe some examples of shields that might be of interest to you.

TinkerKit Breakout Shield

For many of the examples in the book, we have chosen to use the TinkerKit breakout shield (Figure 2.8). It consists of a shield that maps the available pins on the Arduino Mega ADK board in a different way. It unfolds all the pins on single molex connectors with independent power and ground for each. This is useful when you want (or are looking for) a mechanically safe way to connect things to your board.

Figure 2.8. TinkerKit breakout shield

At the other end, the sensors and actuators (Figure 2.9) that can be used with the TinkerKit shield offer the same types of connectors. You are not required to use this shield or the other tools— you can build all the examples on a breadboard with discrete components. It just takes a little longer.

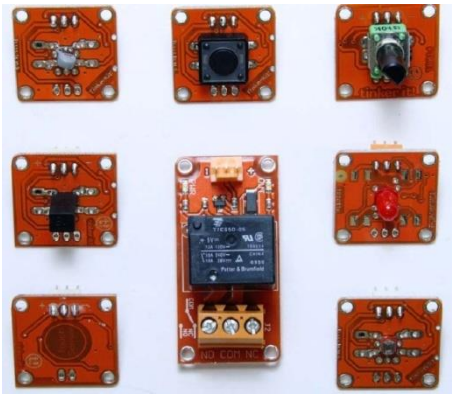

Figure 2.9. Examples of TinkerKit components

USB-Host Shield

The USB Host Shield (Figure 2.10) is a breakout board for the MAX-3421, a chip that can handle the USB protocol to connect any kind of USB device. You can connect a keyboard, mouse, Bluetooth dongle, a 3G modem, or any other USB client and control it from an Arduino Uno or Arduino Mega2560. The Arduino Mega ADK has this very same chip on board, which renders this shield unnecessary in that case.

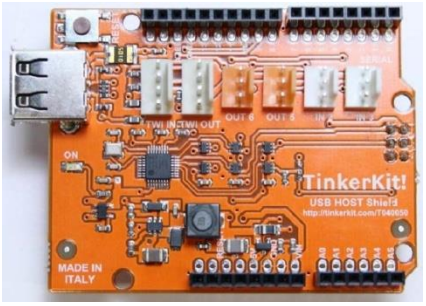

Figure 2.10. USB-Host Shield

Remember that, on top of the shield, you will need to install some libraries to make it work. The code to control the MAX-3421 doesn't come by default with the Arduino IDE.

Motor Shield

Many of the projects coming to life focus on making objects move. Many different types of motors exist and each type requires its own way for driving it. A good approach to control both DC and stepper motors is the Arduino Motor shield, shown in Figure 2-11. With it you can either use four solenoids, or control the speed and direction of two DC motors or one stepper as well as measure the motors' feedback current (a feature broadly used in robotics).

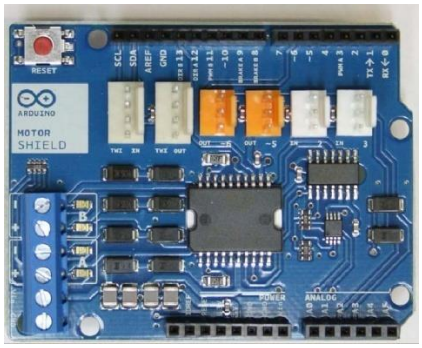

Figure 2.11 Arduino Motor shield

In the figure you can also see the characteristic TinkerKit connectors. Moving objects require better ways to secure the sensors, and having these types of connectors will only make things easier.

2.2 Data communication boards based on the Arduino

The act of communicating requires having two or more parties exchanging, requesting, sending, and evaluating data. Those involved in the information exchange can be people or machines. Successful exchanges require the use of a predetermined mechanism on how to request data, but also on how to acknowledge the arrival of it. The definition of those mechanisms is made in an abstract way and is independent of the transmission channel. It's what we call a protocol.

Different protocols accommodate different scenarios of use. Trying to send data over a 6.000 Km long submarine cable is not the same as using a twisted pair of copper wires between two circuits at 10 cms distance from each other.

Understanding how communication works between two electronic devices requires thinking beyond the bits and electronic components themselves. You need to consider factors like noise, whether the communication happens over wires or in a wireless way, how far the devices are from each other, or how quickly you want data to be sent to the other side.

Sometimes your envisioned application cannot be achieved because of one of the factors is too limiting, but standards exist that can help you get your project done in almost any case.

At the lowest logical level, information is encoded in packages of bits. Most of the existing communication systems use packages of 8 bits (which is the same as 1 byte) as the basic unit for information transfer. Those bytes contain the data you want to send, like the temperature measured by a sensor.

Probably the most obvious way to get your devices to communicate with the physical world is to get it to talk to a connected object. You could use your Arduino hooked to a series of shields that would offer connectivity to some sort of network.

Among others, you could use:

➤ An Arduino Ethernet Shield (Figure 2.12) or equivalent. These boards enable you to connect to a wired network and connect to a server to post data, or even create a small server to which you could connect with your Android device

via a browser. An equivalent use scenario would be using an Arduino Ethernet board, which merges an Arduino Uno together with an Ethernet Shield into a single circuit.

Figure 2.12. Arduino Uno with an Ethernet Shield

➤ An Arduino GSM/GPRS Shield (Figure 2.13) or compatible. With this you can connect to the Internet to post data to servers. Again, you could connect to the data posted by the board by sending requests to the server. It would also be possible to send data from the phone to the board via the intermediating server.

Figure 2.13. Arduino Uno with an GSM/GPRS Shield

➤ An Arduino Wi-Fi Shield (Figure 2.14). It is completely equivalent to the Arduino Ethernet Shield case, but operates over a Wi-Fi connection. In the

same way as with the Ethernet Shield, you do not necessarily need a server between the Android device and your Arduino board. One of them could operate as server and the other as a client in a typical TCP/IP connection.

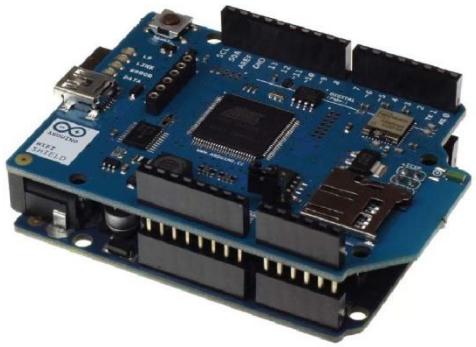

Figure 2.14. Arduino Uno with a Wi-Fi Shield

2.3 Programming Arduino based systems

We learn programming concepts of Arduino based embedded systems based on the LED project. In this project we are going to build and test Arduino program, which is to blink an LED. However, this time we are going to use on of the LED's in the kit and we will also learn about some electronics and coding in C along the way.

Now, first make sure that your Arduino is powered off. You can do this either by unplugging the USB cable or by taking out the Power Selector Jumper on the Arduino board. Then connect everything up like this figure.

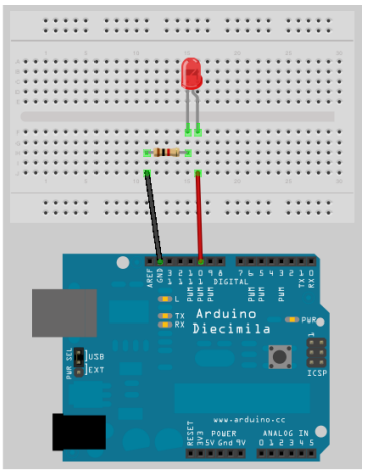

Figure 2.15. Arduino LED project

It doesn't matter if you use different coloured wires or use different holes on the breadboard as long as the components and wires are connected in the sameorder as the picture. Be careful when insertingcomponents into the Breadboard. The Breadboard is brand new and the grips in the holes will be stiff to begin with. Failure to insert components carefully could result in damage.

Make sure that your LED is connected the right way with thelonger leg connected to Digital Pin 10. The long led is the Anode of the LED and always must go to the +5v supply (in this case coming out of Digital Pin 10) and the short leg is the Cathode and must go to Gnd (Ground).

When you are happy that everything is connected up correctly, power up your Arduino and connect the USB cable.

Now, open up the Arduino IDE and type in the following code:

```
intledPin = 10;
void setup()
{
pinMode(ledPin, OUTPUT);
}
void loop()
```

```
    {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

Now press the Verify/Compile button at the top of the IDE to make sure there are no errors in your code. If this is successful you can now click the Upload button to upload the code to your Arduino.

If you have done everything right you should now see the Red LED on the breadboard flashing on and off every second.

Now let's take a look at the code and the hardware and find out how they both work.

So let's take a look at the code for this project. Our first line is

```
// Project 1 - LED Flasher
```

This is simply a comment in your code and is ignored by the compiler (the part of the IDE that turns your code into instructions the Arduino can understand before uploading it). Any text entered behind a // command will be ignored by the compiler and is simply there for you, or anyone else that reads your code. Comments are essential in your code to help you understand what is going on and how your code works. Comments can also be put after commands as in the next line of the program.

Later on as your projects get more complex and your code expands into hundreds or maybe thousands of lines, comments will be vital in making it easy for you to see how it works. You may come up with an amazing piece of code, but if you go back and look at that code days, weeks or months alter, you may forget how it all works. Comments will help you understand it easily. Also, if your code is meant to be seen by other people (and as the whole ethos of the Arduino, and indeed the whole Open Source, community is to share code and schematics,

we hope when you start making your own cool stuff with the Arduino you will be willing to share it with the world) then comments will enable that person to understand what is going on in your code.

You can also put comments into a block statement by using the /* and */ commands. E.g.

The next line of the program is

```
intledPin = 10;
```

This is what is known as a variable. A variable is a place to store data. In this caseyou are setting up a variable of type int or integer. An integer is a number within the range of -32,768 to 32,767. Next you have assigned that integer the name of ledPin and have given it a value of 10. We didn't have to call it ledPin, we could have called it anything we wanted to. But, as we want our variable name to be descriptive we call it ledPin to show that the use of this variable is to set which pin on the Arduino we are going to use to connect our LED. In this case we are using Digital Pin 10. At the end of this statement is a semi-colon. This is a symbol to tell the compiler that this statement is now complete.

Although we can call our variables anything I want, every variable name in C must start with a letter, the rest of the name can consist of letters, numbers and underscore characters. C recognizes upper and lower case characters as being different. Finally, you cannot use any of C's keywords like main, while, switch etc as variable names. Keywords are constants, variables and function names that are defined as part of the Arduino language. Don't use a variable name that is the same as a keyword. All keywords within the sketch will appear in red.

So, you have set up an area in memory to store a number of type integer and have stored in that area the number 10. Imagine a variable as a small box where you can keep things. A variable is called a variable because you can change it. Later on we will carryout mathematical calculations on variables to make our program do more advanced stuff.

Next we have our setup() function

```
void setup()
```

```
{
pinMode(ledPin, OUTPUT);
}
```

An Arduino sketch must have a setup() and loop() function otherwise it will not work. The setup() function is run once and once only at the start of the program and is where you will do issue general instructions to prepare the program before the main loop runs, such as setting up pin modes, setting serial baud rates, etc.

Basically a function isa block of code assembled into one convenient block. For example, if we created our own function to carry out a whole series of complicated mathematics that had many lines of code, we could run that code as many times as we liked simply by calling the function name instead of writingout the code again each time. Later on we will go into functions in more detail when we start to create our own.

In the case of our program the setup() function only has one statement to carry out. The function starts with

```
Void setup()
```

and here we are telling the compiler that our function is called setup, that it returns no data (void) and that we pass no parameters to it (empty parenthesis). If our function returned an integer value and we also had integer values to pass to it (e.g. For the function to process) then it would look something like this

```
intmyFunc(int x, int y)
```

In this case we have created a function (or a block of code) called myFunc. This function has been passed two integers called X and Y. Once the function has finished it will then return an integer value to the point after where our function was called in the program (hence int before the function name).

All of the code within the function is contained within the curly braces. A {symbol starts the block of code and a} symbol ends the block. Anything in between those two symbols is code that belongs to the function.

We will go into greater detail about functions later on so don't worry about them for now. All you need to know is that in this program, we have two functions, the first function is called setup and it's purpose is to setup anything necessary for our program to work before the main program loop runs.

```
void setup()
{
pinMode(ledPin, OUTPUT);
}
```

Our setup function only has one statement and that is pinMode. Here we are telling the Arduino that we want to set the mode of one of our digital pins to be Output mode, rather than Input. Within the parenthesis we put the pin number and the mode (OUTPUT or INPUT).

Our pin number is ledPin, which has been previously set to the value 10 in our program. Therefore, this statement is simply telling the Arduino that the Digital Pin 10 is to be set to OUTPUT mode.

As the setup() function runs only once, we now move onto the main function loop.

```
void loop()
{
digitalWrite(ledPin, HIGH);
delay(1000);
digitalWrite(ledPin, LOW);
delay(1000);
}
```

The loop() function is the main program function and runs continuously as long as our Arduino is turned on. Every statement within the loop() function (within the curly braces) is carried out, one by one, step by step, until the bottom of the function is reached, then the loop starts again at the top of the function, and so on forever or until you turn the Arduino off or press the Reset switch.

In this project we want the LED to turn on, stay on for one second, turn off and remain off for one second, and then repeat. Therefore, the commands to tell the Arduino to do that are contained within the loop () function as we wish them to repeat over and over.

The first statement is

Digital Write (ledPin, HIGH);

This writes a HIGH or a LOW value to the digital pin within the statement (in this case ledPin, which is Digital Pin 10). When you set a digital pin to HIGH you are sending out 5 volts to that pin. When you set it to LOW the pin becomes 0 volts, or Ground.

This statement therefore sends out 5v to digital pin 10 and turns the LED on.After that is

delay (1000);

and this statement simply tells the Arduino to wait for 1000 milliseconds (to 1 second as there are 1000 milliseconds in a second) before carrying out the next statement which is

digitalWrite(ledPin, LOW);

which will turn off the power going to digital pin 10 and therefore turn the LED off. There is then another delay statement for another 1000 milliseconds and then the function ends. However, as this is our main loop() function, the function will now start again at the beginning. By following the program structure step by step again we can see that it is very simple.

```
// Project 1 - LED Flasher
intledPin = 10;
void setup()
{
pinMode(ledPin, OUTPUT);
}
void loop()
```

```
{
digitalWrite(ledPin, HIGH);
delay(1000);
digitalWrite(ledPin, LOW);
delay(1000);
}
```

We start off by assigning a variable called ledPin, giving that variable a value of 10.

Then we move onto the setup() function where we simply set the mode for digital pin 10 as an output.

In the main program loop we set Digital Pin 10 to high, sending out 5v. Then we wait for a second and then turn off the 5v to Pin 10, before waiting another second. The loop then starts again at the beginning and the LED will therefore turn on and off continuously for as long as the Arduino has power.

Now that you know this you can modify the code to turn the LED on for a different period of time and also turn it off for a different time period.

For example, if we wanted the LED to stay on for 2 seconds, then go off for half a second we could do this:

```
void loop()
{
digitalWrite(ledPin, HIGH);
delay(2000);
digitalWrite(ledPin, LOW);
delay(500);
}
```

or maybe you would like the LED to stay off for 5 seconds and then flash briefly (250ms), like the LED indicator on a car alarm then you could do this:

```
void loop()
{
digitalWrite(ledPin, HIGH);
```

```
  delay(250);
  digitalWrite(ledPin, LOW);
  delay(5000);
  }
```

or make the LED flash on and off very fast

```
void loop()
{
digitalWrite(ledPin, HIGH);
delay(50);
digitalWrite(ledPin, LOW);
delay(50);
}
```

By varying the on and off times of the LED you create any effect you want. Well, within the bounds of a single LED going on and off that is.

Before we move onto something a little more exciting let's take a look at the hardware and see how it works.

However, let us take a look at a few new keywords and concepts. Here we have a new data types for a variable.

Previously we have created integer data types, which can store a number between -32,768 and 32,767. This time we have created a data type of long, which can store a number from -2,147,483,648 to 2,147,483,647. However, we have specified an unsigned long, which means the variable cannot store negative numbers, which gives us a range from 0 to 4,294,967,295. If we were to use an integer to store the length of time since the last change of lights, we would only get a maximum time of 32 seconds before the integer variable reached a number higher than it could store.

You may well ask why we don't just have one data type that can store huge numbers all the time and be done with it. Well, the reason we don't do

that is because variables take up space in memory and the larger the number the more memory is used up for storing variables. On your home PC or laptop you won't have to worry about that much at all, but on a small microcontroller like the Atmega328 that the Arduino uses it is essential that we use only the smallest variable data type necessary for our purpose.

There are various data types that we can use as our sketches.

Table 2.2. Data types in Arduino

Data type	RAM	Number Range
void keyword	N/A	N/A
boolean	1 byte	0 to 1 (True or False)
byte	1 byte	0 to 255
char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
int	2 byte	-32,768 to 32,767
unsigned int	2 byte	0 to 65,535
word	2 byte	0 to 65,535
Long	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long	4 byte	0 to 4,294,967,295
float	4 byte	-3.4028235E+3 to 3.4028235E+38
double	4 byte	-3.4028235E+3 to 3.4028235E+38
String	1 byte+ x	Arrays of chars
array	1 byte + x	Collection of variables

Each data type uses up a certain amount of memory on the Arduino as you can see on the chart above. Some variables use only 1 byte of memory and others use 4 or more (don't worry about what a byte is for now as we will discuss this later). You cannot copy data from one data type to another, e.g. If x was an int and y was a stringthen x = y would not work as the two data types are different.

The Atmega168 has 1Kb (1000 bytes) and the Atmega328 has 2Kb (2000 bytes) of SRAM. This is not a lot and in large programs with lots of

variables you could easily run out of memory if you do not optimize your usage of the correct data types. From the list above we can clearly see that our use of the int data type is wasteful as it uses up 2 bytes and can store a number up to 32,767. As we have used int to store the number of our digital pin, which will only go as high as 13 on our Arduino (and up to 54 on the Arduino Mega), we have used up more memory than was necessary. We could have saved memory by using the byte data type, which can store a number between 0 and 255, which is more than enough to store the number of an I/O pin.

Next we have

```
pinMode(button, INPUT);
```

This tells the Arduino that we want to use Digital Pin 2 (button = 2) as in INPUT. We are going to use pin 2 to listen for button presses so it's mode needs to be set to input.

In the main program loop we check the state of digital pin 2 with this statement:

```
int state = digitalRead(button);
```

This initializes an integer(yes it's wasteful and we should use a boolean) called 'state' and then sets the value of state to be the value of the digital pin 2. The digitalRead statement reads the state of the digital pin within the parenthesis and returns it to the integer we have assigned it to. We can then check the value in state to see if the button has been pressed or not.

```
if (state == HIGH && (millis() - changeTime) > 5000)
{
   // Call the function to change the lights
```

changeLights();
}

The if statement is an example of a control structure and it's purpose is to check if a certain condition has been met or not and if so to execute the code within it's code block. For example, if we wanted to turn anLED on if a variable called x rose above the value of 500 we could write

if (x>500) {digitalWrite(ledPin, HIGH);

When we read a digitalpin using the digitalRead command, the state of the pin will either be HIGH or LOW. So the if command in our sketch looks like this

if (state == HIGH && (millis() - changeTime) > 5000)

What we are doing here is checking that two conditions have been met. The first is that the variable called state is high. If the button has been pressed state will be high as we have already set it to be the value read in from digital pin 2. We are also checking that the value of millis()-changeTime is greater than 5000 (using the logical AND command &&). The millis() function is one built into the Arduino language and it returns the number of milliseconds since the Arduino started to run the current program. Our changeTime variable will initially hold no value, but after the changeLights) function has ran we set it at the end of that function to the current millis() value.

By subtracting the value in the changeTime variable from the current millis() value we can check if 5 seconds have passed since changeTime was last set. The calculation of millis()-changeTime is put inside it's own set of parenthesis to ensure that we compare the value of state and the result of this calculation and not the value of millis() on its own.

The symbol '&&' in between

state == HIGH

and the calculation is an example of a Boolean Operator. In this case it means AND. To see what we mean by that, let's take a look at all of the Boolean Operators.

&&$ Logical AND

||$ Logical OR

!$ NOT

These are logic statements andcan be used to test various conditions in if statements.

&& means true if both operands are true, e.g.:

if (x==5 && y==10) {....

This if statement will run it's code only if x is 5 and also y is 10.

|| means true if either operand is true, e.g. :

if (x==5 || y==10) {.....

This will run if x is 5 or if y is 10.The! or NOT statement means true if the operand is false, e.g. :

if (!x) {.....

Will run if x is false, i.e. equals zero.You can also 'nest' conditions with parenthesis, for example

if (x==5 && (y==10 || z==25)) {.....

In this case, the conditions within the parenthesis are processed separately and treated as a single condition and then comparedwith the second condition. So, if we draw a simple truth table for this statement we can see how it works.

Table 2.3. Thecondition statement

x	y	z	True/False?
4	9	25	FALSE
5	10	24	TRUE
7	10	25	FALSE
5	10	25	TRUE

The main program loop simply checks continuously if the pedestrian button has been pressed or not and if it has, and (&&) the time since the lights were last changed is greater than 5 seconds, it calls the function again.

SUMMARY

You can choose from many platforms, but for simplicity you work with only one or two in this book. At the same time, many vendors make chips. Arduino's approach is to have a vendor-independent software core to enable you to move from platform to platform seamlessly.

Shields are an easy way to bring sensors and actuators into your projects, minimizing the amount of soldering required to get things done. You can find shields for Arduino that do almost anything.

One important issue to write program for hardware systems. For coding embedded systems based on the Arduino we use C language. Specialized C for Arduino is defined above.

3. DESIGNING EMBEDDED COMMUNICATION SYSTEM BY USING DIGITAL INTERFACE OF ARDUINO UNO PLATFORM

3.1 Digital Inputs/Outputs and control on the Arduino Uno platform

The digital pins 0 to 12 can all be used as either an input or an output. Since you are going to be connecting electronics to one of these pins, it is unlikely that you are going to want to change the mode of a pin. That is, once a pin is set to be an output, you are not going to change it to be an input midway through a sketch.

For this reason, it is a convention to set the direction of a digital pin in the setup function that must be defined in every sketch.

For example, the following code sets digital pin 10 to be an output and digital pin 11 to be an input. Note how we use a variable declaration in our sketch to make it easier to change the pin used for a particular purpose later on.

```
intledPin = 10;
intswitchPin = 11;
void setup()
{
pinMode(ledPin, OUTPUT);
pinMode(switchPin, INPUT);
}
```

So now we know how to set a digital pin to be an input, we can build a project for model traffic signals using red, yellow, and green LEDs. Every time we press the button, the traffic signal will go to the next step in the sequence. In the Uzb, the sequence of such traffic signals is red, red and amber together, green, amber, and then back to red.

As a bonus, if we hold the button down, the lights will change in sequence by themselves with a delay between each step.

The components for Project are listed next. When using LEDs, for best effect, try and pick LEDs of similar brightness.

COMPONENTS AND EQUIPMENT

	Description	Appendix
	Arduino Diecimila or Duemilanove board or clone	1
D1	5-mm red LED	23
D2	5-mm yellow LED	24
D3	5-mm green LED	25
R1-R3	270 Ω 0.5W metal film resistor	6
R4	100 KΩ 0.5W metal film resistor	13
S1	Miniature push to make switch	48

Figure 3.1. The project components and equipment

The schematic diagram for the project is shown in Figure 3.1.

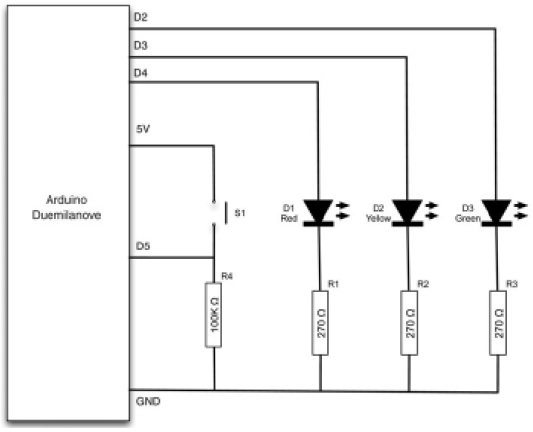

Figure 3.2. Schematic diagram of project

The LEDs are connected in the same way as our earlier project, each with a current-limiting resistor. The digital pin 5 is “pulled” down to GND by R4 until the switch is pressed, which will make it go to 5V.

Software

The sketch for Project is shown in Listing Project. The sketch is fairly self-explanatory. We only check to see if the switch is pressed once a second, so pressing the switch rapidly will not move the light sequence on. However, if we press and hold the switch, the lights will automatically sequence round.

We use a separate function set Lights to set the state of each LED, reducing three lines of code to one.

Listing project

```
intredPin = 2;
intyellowPin = 3;
intgreenPin = 4;
intbuttonPin = 5;
int state = 0;
    void setup()
    {
    pinMode(redPin, OUTPUT);
    pinMode(yellowPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(buttonPin, INPUT);
    }
    void loop()
    {
    if (digitalRead(buttonPin))
    {
    if (state == 0)
    {
    setLights(HIGH, LOW, LOW);
```

```
state = 1;
}
else if (state == 1)
{
setLights(HIGH, HIGH, LOW);
state = 2;
}
else if (state == 2)
{
setLights(LOW, LOW, HIGH);
state = 3;
}
else if (state == 3)
{
setLights(LOW, HIGH, LOW);
state = 0;
}
delay(1000);
}
}
voidsetLights(int red, int yellow,
int green)
{
digitalWrite(redPin, red);
digitalWrite(yellowPin, yellow);
digitalWrite(greenPin, green);
}
```

Test the project by holding down the button and make sure the LEDs all light in sequence.

3.2 Designing of Embedded Communication systemfor control access

With digital interfaces we can design and develop different type of communication systems. These embedded communication systems works with different objects on the control, monitoring and management purposes. Because of these, we design communication system for control access. For this purpose, we use Arduino embedded board, RFID module and control object. Schematic diagram of the project is given below (Figure 3.3)

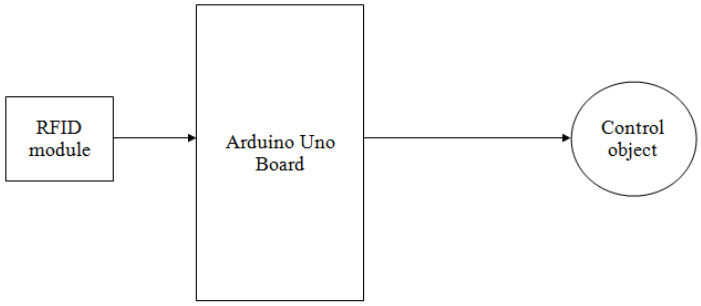

Figure 3.3. Schematic diagram of the embedded control system

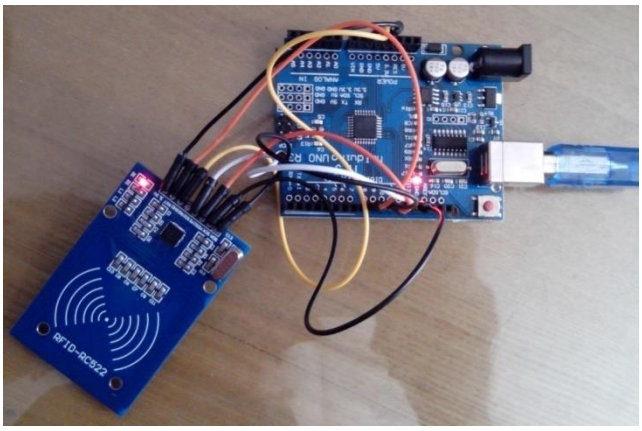

Figure 3.4. Arduino and RFID project

We develop code and check and compile with Arduino IDE. Listing for the project is given below.

Listing 3.1

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.
MFRC522::MIFARE_Key key;
/**
 * Initialize.
 */
void setup()
{
Serial.begin(9600); // Initialize serial communications with the PC
while (!Serial);    // Do nothing if no serial port is opened
SPI.begin();        // Init SPI bus
   mfrc522.PCD_Init(); // Init MFRC522 card
   // Prepare the key (used both as key A and as key B)
   // using FFFFFFFFFFFFh which is the default at chip delivery from the factory
for (byte i = 0; i < 6; i++) {
key.keyByte[i] = 0xFF;
   }
Serial.println(F("Scan a MIFARE Classic PICC to demonstrate read and write."));
Serial.print(F("Using key (for A and B):"));
dump_byte_array(key.keyByte, MFRC522::MF_KEY_SIZE);
```

```
Serial.println();
Serial.println(F("BEWARE: Data will be written to the PICC, in sector #1"));
}
  /**
   * Main loop.
   */
  void loop() {
   // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent())
  return;
// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial())
return;
    // Show some details of the PICC (that is: the tag/card)
Serial.print(F("Card UID:"));
dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
Serial.println();
Serial.print(F("PICC type: "));
bytepiccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccType));
    // Check for compatibility
if (    piccType != MFRC522::PICC_TYPE_MIFARE_MINI
&&piccType != MFRC522::PICC_TYPE_MIFARE_1K
&&piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
Serial.println(F("This sample only works with MIFARE Classic cards."));
return;
    }
    // In this sample we use the second sector,
```

```
    // that is: sector #1, covering block #4 up to and including block #7
  byte sector        = 1;
  byteblockAddr      = 4;
  bytedataBlock[]    = {
      0x01, 0x02, 0x03, 0x04, //  1,  2,   3,  4,
      0x05, 0x06, 0x07, 0x08, //  5,  6,   7,  8,
      0x08, 0x09, 0xff, 0x0b, //  9, 10, 255, 12,
      0x0c, 0x0d, 0x0e, 0x0f  // 13, 14,  15, 16
    };
  bytetrailerBlock   = 7;
  byte status;
  byte buffer[18];
  byte size = sizeof(buffer);
    // authenticate using key A
  Serial.println(F("Authenticating using key A..."));
  status =
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock,
&key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
  Serial.print(F("PCD_Authenticate() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
   }
    // Show the whole sector as it currently is
  Serial.println(F("Current data in sector:"));
    mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key,
sector);
  Serial.println();
```

```
   // Read data from the block
Serial.print(F("Reading data from block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
status = mfrc522.MIFARE_Read(blockAddr, buffer, &size);
if (status != MFRC522::STATUS_OK) {
Serial.print(F("MIFARE_Read() failed: "));
Serial.println(mfrc522.GetStatusCodeName(status));
   }
Serial.print(F("Data in block ")); Serial.print(blockAddr); Serial.println(F(":"));
dump_byte_array(buffer, 16); Serial.println();
Serial.println();
   // Authenticate using key B
Serial.println(F("Authenticating again using key B..."));
  status =
  mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B,
  trailerBlock, &key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
Serial.print(F("PCD_Authenticate() failed: "));
Serial.println(mfrc522.GetStatusCodeName(status));
return;
   }
   // Write data to the block
Serial.print(F("Writing data into block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
dump_byte_array(dataBlock, 16); Serial.println();
status = mfrc522.MIFARE_Write(blockAddr, dataBlock, 16);
if (status != MFRC522::STATUS_OK) {
Serial.print(F("MIFARE_Write() failed: "));
```

```
        Serial.println(mfrc522.GetStatusCodeName(status));
           }
        Serial.println();
           // Read data from the block (again, should now be what we have written)
        Serial.print(F("Reading data from block ")); Serial.print(blockAddr);
        Serial.println(F(" ..."));
        status = mfrc522.MIFARE_Read(blockAddr, buffer, &size);
        if (status != MFRC522::STATUS_OK) {
        Serial.print(F("MIFARE_Read() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
           }
        Serial.print(F("Data in block ")); Serial.print(blockAddr); Serial.println(F(":"));
        dump_byte_array(buffer, 16); Serial.println();
           // Check that data in block is what we have written
           // by counting the number of bytes that are equal
        Serial.println(F("Checking result..."));
        byte count = 0;
        for (byte i = 0; i < 16; i++) {
              // Compare buffer (= what we've read) with dataBlock (= what we've
written)
        if (buffer[i] == dataBlock[i])
        count++;
           }
        Serial.print(F("Number of bytes that match = ")); Serial.println(count);
        if (count == 16) {
        Serial.println(F("Success :-)"));
           } else {
        Serial.println(F("Failure, no match :-("));
```

```
        Serial.println(F("  perhaps the write didn't work properly..."));
           }
        Serial.println();
          // Dump the sector data
        Serial.println(F("Current data in sector:"));
           mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key,
sector);
        Serial.println();
           // Halt PICC
           mfrc522.PICC_HaltA();
           // Stop encryption on PCD
           mfrc522.PCD_StopCrypto1();
        }
        /**
         * Helper routine to dump a byte array as hex values to Serial.
        */
        voiddump_byte_array(byte *buffer, byte bufferSize) {
        for (byte i = 0; i <bufferSize; i++) {
        Serial.print(buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(buffer[i], HEX);
        }
        }
```

After compiling code, we check data sent results with ID card (figure 3.4) and control object codes via com emulator(figure 3.5).

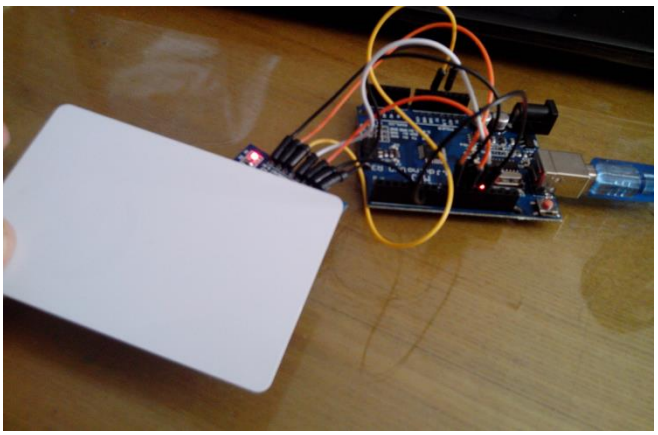

Figure 3.4. ID card communication with Arduino board

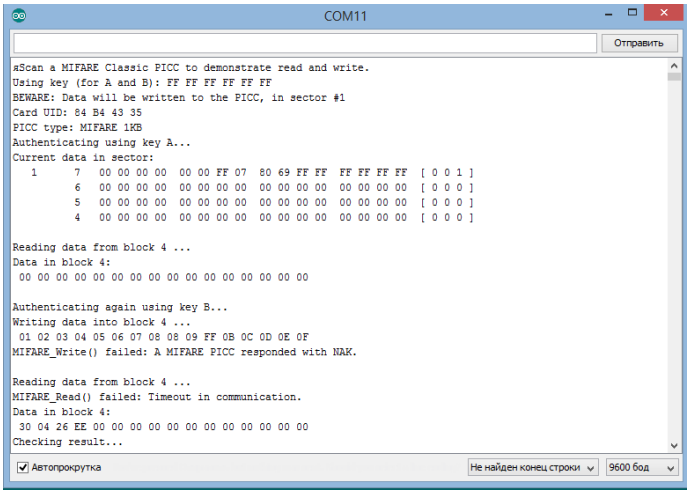

Figure 3.5. Com emulators with RFID data send results

SUMMARY

In this chapter I design embedded communication control system based on the arduino board's digital communication interface. RFID module, arduino and control object are main blocks of project. This system is used for control and register objects via digital interface.For these purposes we use wired and wireless digital communication lines.

The source code of project is written in C and compiled with Arduino IDE. The project is tested on the real environment.

4. LIFE SAFETY AND ECOLOGIYA

4.1 Safety requirements for the construction and installation of communications equipment works

The production rooms unpacking the equipment is prohibited. Unpacking should be done in a room located adjacent to or near the production areas. To extract can be used hallway or another room.

In carrying out construction works should be used with power voltage 42 V when working at height using ladders. Servicing and maintenance of electrical installations is prohibited to use metal ladders.

All work must be carried out according to the document "Rules for Safety in the work on the telephone and telegraph stations"

To work at height are allowed persons who are allowed to work at height. Work on designs that do not have fences, as well as work related to going beyond the fences, climbers must be carried out. To separate Steeplejacks allowed, the person (workers, engineers and technicians) are at least 18 years have passed medical examination and found fit, having the experience of steeplejack work for at least one year and tariff rank not lower than the third.

Steeplejack considered work performed at a height of 5 m above the ground, floor or working platform on which work is carried directly to the structures during installation or repair, and the primary means which protects workers from falling from a height, a safety belt.

When working on a pedestal in the danger zone is allowed to enter only persons directly connected with the work, provided the use of protective helmets. Danger zone around the masts and towers is determined by operation and maintenance center distance from the support (mast tower) equal to 1/3 of its height.

The rise of people on the antenna mast structures are prohibited:

- not withdrawn at voltages greater than 42 V;

- during a lightning storm and in its approach;
- black ice, heavy rain, snow or fog;
- when not fastened to the cradle carbine safety belt;
- at wind speeds above 12 m / s, in the dark;
- the lifting device, the term of each test has expired;
- for defective ropes, with a faulty winch;
- without a safety helmet and safety belt.

Those working on the mast and the antenna is allowed to perform repair work on the support only after he stuck to her designs with a safety belt. Paint should support from ladders, scaffolds, or bassinet.

Employees of the mast and the antenna must wear protective helmets and protective footwear (boots, shoes without metal nails and with non-slip soles).

Work on the AMC must be carried by at least two employees of the mast, one of whom is watching. The observer must be the center of the pole at a distance no closer than one-third of its height and carry Fitters belt, and in the case of works on wooden poles - and claws.

Welding work must be carried to the inventory cradle hoist provided suspension cradles to the ropes.

During a thunderstorm, and as it approached prohibited from being around grounding. The work on the antenna field must stop, and people move into the premises. On the places of grounding must be installed warning signs, "Stop! Voltage ".

4.2. Safety of workplace

Protection against lightning - a whole set of technical solutions and special devices. Protection molniinuzhna to protect against a direct lightning strike to the building, protecting it from the secondary manifestations, such as over-voltage (crosstalk arising in electrical circuits by lightning discharges).

Lightning protection is divided into external and internal.

External lightning protection is a system that ensures interception of lightning and discharge it into the ground, thereby protecting it from damage and fire.

Internal lightning protection is a set of measures and devices (SPD) designed for bonding: eliminate the possibility of a dangerous voltage in electrical circuits and piping within the building.

The composition of external lightning protection:

The receivers of lightning - a device that captures a bolt of lightning (lightning rod)

Current conductors (downhill) is part of the lightning protection system, designed to divert lightning current lightning from the receiver to the earth

The earthing - metallic conductor buried in the soil, providing the spreading of the lightning current into the ground.

Internal lightning protection devices consist of surge protection (SPD) and an effective earthing system.

Electrical safety - system of organizational and technical measures and means to protect people from harmful and dangerous effects of electric current, electric arc, electromagnetic fields and static electricity.

Maintenance of electrical installations entrusted to individuals undergoing medical examination and special education.

First aid in case of accidents on the electric current consists of two phases: the release of the victim of the current action and the provision of some medical attention.

The main measures of protection against electric shock are:

1. ensure inaccessibility of live parts under voltage, for a casual touch, electrical separation of the network;

2. eliminating the risk of the appearance of stress on the hulls and other parts of the electrical equipment, which is achieved by the use of low-voltage, using a double isolation, the potential equalization, protective earth, safety off.

The inaccessibility of live parts of electrical installations for accidental contact can be achieved a number of ways: insulated live parts, placing them on the inaccessible height fence.

Under fire safety is understood to mean the state of the object, at which very likely prevented the possibility of fire, and if it occurs the effective protection of people from the dangerous and harmful factors of fire and rescue wealth.

Fire safety of industrial facilities provided by the design and implementation of systems, fire prevention and fire protection systems. The rooms should be equipped with fire-fighting appliances for extinguishing (fire extinguishers). This problem is solved as in the design stage of equipment, and during its operation.

The factors that cause a fire are computers and other electrical appliances. A very common source of fire is smoking in unauthorized places and careless handling of fire personnel. Distribution and sources of ignition are associated with the use of electrical energy. This is primarily a short-circuit which are accompanied by high heat, in the form of arc fault zone with a sprinkling of metal.

Since in this case fires electrical equipment may be energized, the use of water and foam extinguishing unacceptable because it can lead to electrical injury. Therefore, for fighting fires in this room can be used either dry chemical or carbon dioxide extinguishing installation. However, since the latter are intended only to put out small fires, the scope of their application is limited. Therefore, for extinguishing fires in this case, the powder formulations because they have the

following properties: dielectrics, substantially non-toxic and do not have a corrosive effect on metals, do not destroy the insulating varnishes.

The fire extinguishing installation can be both portable and stationary, and can be fixed with manual, remote and automatic switching.

4.3 Electromagnetic radiation and methods of protection of human health

Among the various physical factors of the environment that may have an adverse effect on human and biological objects are more complex nature of non-ionizing electromagnetic fields especially those related to RF energy.

Electromagnetic fields - a special form of existence of matter characterized by a set of electric and magnetic properties. The main parameters that characterize the electromagnetic field are frequency, wavelength, and the propagation velocity.

Natural sources of electromagnetic fields are divided into two groups. The first - the field of the Earth - a constant electric and constant magnetic field. The second group - the radio waves generated by cosmic sources (the sun, stars, etc.), atmospheric processes - lightning bolts, etc.

Anthropogenic sources are divided into 2 groups:

1. Sources of low-frequency radiation (0 - 3 kHz). This group includes all the systems of production, transmission and distribution of electricity (power lines, transformer substations, power plants, various cable systems), home and office electrical and electronic equipment, including computer monitors, transport on the drive, w / d transport and its infrastructure, as well as the subway, trolley and tram transport.

2. Sources of high-frequency radiation (from 3 kHz to 300 GHz). This group includes functional transmitters - sources of electromagnetic fields in order

to transmit or receive information. It is the commercial broadcasters (radio and television), radio (auto and cordless phones, CB radios, amateur radios, cordless phones manufacturing) directed radio (satellite radio, terrestrial relay stations), Navigation (air traffic, navigation, a radio), radars (air communication, navigation, transportation locators, control of air transport). This includes various technological equipment using microwave radiation, variable (50 Hz - 1 MHz) and pulsed fields, household equipment (microwave oven), image display on a cathode-ray tubes (PC monitors, televisions, etc.) .

The extent of the biological effects of electromagnetic fields on the human body depends on the vibration frequency, intensity, and the intensity of the field, its mode of generation (pulsed and continuous), duration of exposure.

Fields microwaves can affect the eyes, leading to cataracts (clouding of the lens), and moderate - to change the type of retinal angiopathy .

After a long stay in range of the electromagnetic fields to premature fatigue, drowsiness or insomnia, there are frequent headaches, there comes a disorder of the nervous system, etc.

Multiple repeated irradiation of low intensity can lead to persistent functional disorders of the central nervous system, persistent neuropsychiatric diseases, changes in blood pressure, slowing heart rate, trophic phenomena (hair loss, brittle nails, etc.).

Protection of the human body from the effects of electromagnetic radiation involves the reduction of their intensity to levels that do not exceed the maximum allowable. Protection provided by the choice of specific methods and tools based on their economic performance, reliability and ease of use.

There are the following methods to protect people from electromagnetic effects:

1. Protection time. It is used when it is impossible to reduce the intensity of radiation at a given point to the permitted level. By designation, notification, etc. limited time finding people in the area expressed influence of the electromagnetic

field. Existing regulations stipulated relationship between the intensity of the energy density and the irradiation time.

2. Protection distance. It is used when you can not reduce the impact of other measures, including the protection of the times. The method is based on the incidence of the radiation intensity proportional to the square of the distance from the source.

3. Protecting the distance as the basis for the valuation of sanitary protection zones - a necessary gap between the field sources and residential buildings, office space, etc. Zone boundaries are determined by calculations for each case placing the emitting installation when using it to maximum power output.

Engineering measures to protect people from electromagnetic effects. Engineering protective measures are based on the phenomenon of shielding electromagnetic fields, or on the restriction of the field emission source parameters (lower intensity). When this second method is used mainly in the design phase of the radiating object.

To protect the public from exposure to electromagnetic radiation may be sought in building construction: metal mesh, sheet metal or any other conductive coating, as well as a specially designed building materials.

Personal protective equipment is designed to prevent human exposure to electromagnetic radiation levels exceeding the maximum permissible when the use of other means is impossible or impractical. They can provide overall protection or the protection of the individual parts of the body (local protection).

4.4. Psycho physiological load person

In the section of psychophysiological stress the most important is stress and fatigue.

Under stress is understood the emotional state that arises in response to all sorts of extreme exposure.

When stress ordinary emotions are replaced by anxiety, causing disturbances in physiological and psychological terms. This concept was introduced by Hans Selye to refer to non-specific response of the body to any adverse effects. His research showed that the various adverse factors - fatigue, fear, hurt, cold, pain, humiliation, and more in the body cause the same kind of comprehensive response regardless of what kind of stimulus acts on it at the moment. Moreover, these stimuli need not exist in reality. A man reacts not only to the actual danger and the threat or reminder of her.

Human behavior in situations of stress is different from the affective behavior. Under stress a person can usually control their emotions, to analyze the situation, make appropriate decisions.

Currently, depending on the stress factor identify different types of stress, including the pronounced physiological and psychological. Psychological stress, in turn, can be divided into information and emotional. If a person is unable to cope with the problem, do not have time to make the right decisions at the required rate with a high degree of responsibility, ie, when there is information overload may develop informational stress. Emotional stress arises in situations of danger, resentment, etc. Hans Selye identified in the development of stress three phases. The first stage - the alarm reaction - the mobilization phase defenses, which increases the stability with respect to a particular traumatic stress. In this case, there is a redistribution of body reserves: our primary objective is due to minor problems. The second step - the stabilization of parameters derived from the balance in the first phase, fixed at a new level. Externally, the behavior is not very different from the norm, as if everything is adjusted, but internally is overrun adaptive reserves. If the stressful situation persists, there comes the third stage - exhaustion, which can lead to a significant deterioration of health, various diseases, and in some cases death.

Stages of development of the state of stress in humans:

• build-up of tension;

• proper stress;

• Reduction of internal tension.

In its first phase duration is strictly individual. Some people "plant" for 2-3 minutes, and another increase in stress can take place over several days or even weeks. But in any case, the state and behavior of the person who is in stress, change pas' opposite sign. "

So, quiet reserved person becomes fussy and irritable, he may even become aggressive and violent. And the person in real life lively and agile, it becomes dark and taciturn.

In the first stage of stress weakens a person self-control: it gradually loses the ability to knowingly and intelligently regulate their own behavior.

The second stage of the stress state is manifested in the fact that man is a loss of effective self-conscious (full or partial). "The Wave" destructive stress damaging to the human psyche. He can not remember what he said and did, or be aware of their actions, rather vague and incomplete. Many then noted that under stress they have done that in a tranquil setting would not have done. Usually all later regret it very much.

Also, like the first, the second phase in duration strictly individual - from several minutes or hours - several days or weeks. Having exhausted its energy resources (achieving higher voltage observed when a person feels the devastation, fatigue and

Stress conditions significantly affect the activities of man. People with different features of the nervous system to react differently to the same psychological burden. In some people there is increased activity, mobilization of forces, improve business performance. On the other hand, the stress can cause disruption of the sharp reduction of its effectiveness, and total inhibition of inactivity.

Human behavior in a stressful situation depends on many factors, but primarily on the psychological preparation of a person, which includes the ability to quickly assess the situation, the instantaneous orientation skills in unexpected

circumstances, a strong-willed discipline and determination, experience, behavior in similar situations.

Methods of dealing with stress

Stress - the feeling that one experiences when she believes that it can not effectively cope with the situation.

If the situation is causing stress depends on us, a more rational to focus on how to change it. If the situation is not up to us to accept and change your perception, your attitude to this situation.

In most situations, the stress goes through several stages.

1. Phase anxiety. This mobilization of energy resources of the body. Moderate stress useful in this step, it leads to higher efficiency.

2. Phase resistance. This is a balanced spending reserves. Outwardly, everything looks normal, people effectively solves the problems faced by them, but if this step takes too long and is not accompanied by relaxation, then, the body works hard.

3. Phase depletion (distress). Man feels weakness and fatigue, reduced performance, dramatically increases the risk of disease. Short time this can still fight at will, but then the only way to restore power - it's a solid rest.

One of the most common causes of stress - the contradiction between reality and perceptions of man.

Stress response is equally easy to run as real events, and existing only in our imagination. In psychology this is called "the law of the emotional reality of the imagination." As psychologists have calculated, about 70% of our experiences come about events that do not exist in reality, but only in the imagination.

By the development of stress can lead not only negative but also positive life events. When something changes dramatically for the better, the body also reacts to this stress.

Usually, the fatigue is understood the reduction in the workability caused by previous work, which has a temporary character. If it occurs during mental activity, talk about mental fatigue. State of fatigue is manifested in changes physiological

processes, reducing productivity and techno-economic indicators, change in mental status.

Psychologists say that the development of fatigue, the person has a special psyche, which is called the fatigue - a subjective reflection arising ¬ ing processes in the body, leading to fatigue. It appears long before the loss of productivity lies in the fact that there is a special experience painful stress and uncertainty. Manage feels that he could not continue to work properly. Thus there is a disorder of attention - in the development of fatigue, people are easily distracted, becomes sluggish, inactive, or, on the contrary, it appears chaotic mobility insta ¬ bility. There are disturbances in the sensory area - for fatigue changes work receptor, for example, there is a visual fatigue - decreased ability to process information coming through the visual analyzer, with the durationtion manual work is reduced tactile and kinaesthetic sensitivity. Lead to abnormalities in the motor area: a slowing of movements, movements appear haste, rhythm disorder, weakening the accuracy and co-ordination of movements, de-automatization of movements. There are defects in memory and thinking, weakened the will, determination, endurance, self-control. With strong fatigue, somnolence.

Intensity of change depends on the depth of fatigue. For example, significant changes in mental status almost there, and with fatigue all these changes is extremely pronounced.

Due to changes in the mental state of a number of physiologists has isolated ¬ lyat 3 stages of fatigue. Stage 1: When her with the feeling of fatigue significantly, labor productivity is not reduced. Stage 2 - characterized by a significant reduction ¬ tion of labor and severe mental changes. The third stage, which some scholars regard as acute fatigue, accompanied by the expression experience fatigue.

Utation can be physical (musclenym) or neuropsychiatric (central). Both forms of fatigue combined with hard work, and they can not be strictly separated from one another. Heavy physical work leads primarily to muscle fatigue, and enhanced mental functions or monotonous work is tiringtion of central origin. It

should be a clear distinction between exhaustion and fatigue, caused need for sleep.

In addition, determine the primary Utition, which is developing quite rapidly at the beginning of the work shift and is a recognized com ¬ insufficient consolidation of skills, it can be overcome in the process, resulting in an "second wind" - a significant increase workable STI. Secondary, slowly progressive fatigue actually tiringtion, which occurs after about 2.5-3 hours from the beginning of the work shift, and to remove it needs rest.

Fatigue or chronic fatigue - another type of fatigue. It is due to the lack of proper rest between each working day, is regarded as a pathological condition. Manifests the general decline in productivity, increased incidence, the slowdown in the cultural and technical level and skills of running, decreased creativity and mental capacity, changes in the cardiovascular system.

According to K.K Platonov are four degrees of fatigue restarting, lung, and severe, each of which requires appropriate methods of struggle. So, to relieve fatigue suf beginning precisely regulate the regime of work and rest. Mild fatigue optionally sary to wait for release and use it effectively. In marked overworked SRI urgent needs rest, better organized. In severe pereutomtion to treatment.

OVERALL CONCLUSION

In this workwe explored how to make use of digital inputs and outputs in microcontrollers for communication. Beyond making some small examples, you built two projects: a display to show incoming text messages to your phone/tablet and a keyboard to trigger different sound samplers.

One of the examples required hacking a desk lamp and controlling it with Arduino. Therefore you got introduced to some basic concepts about safety. Remember that it is the product of current voltage that can be harmful. Digital technology is usually operating at 5 Volts or less and should not represent a danger in any way. Arduino boards work at 5 Volts. When connecting to something like the desk lamp, which uses between 110 and 220 Volts AC (depending on the country you are in) you need to make sure everything is unplugged before you go on touching the circuits, wires, and so on.

When building prototypes with Arduino it's important to take proper safety measures, not only for your own sake but also for the sake of the components. Some parts are inexpensive, but some others, like accelerometers, gyroscopes, and other complex sensors, break easily. A couple of important things to remember are:

➤We should remember to power your components the right way: Vin means voltage in, GND is 0 V. Some components are powered at 5 Volts, whereas some others work at 3.3 Volts.

➤ Respect the polarity! Some components, like LEDs, will not operate when plugged in wrong. Other components, like resistors, have no polarity.

➤ LEDs operate at a fixed voltage that is less than the V_{in} at your circuit. You will need to use a resistor to protect them from burning.

➤ There exist different protocols to control devices. For example, the LED screen used for the SMS display project uses an SPI protocol for controlling all the LEDs from a minimal set of pins. Other protocols are I2C or Serial — also known as UART.

➤ Mnemotechnic rule comparing voltage values with their binary representation and with the logical representation in code. We have only hinted at this rule throughout the text, but we needed to formulate it at once:

➤ HIGH = V_{in} = Boolean TRUE = logical 1;

➤ LOW = 0 Volts = Boolean FALSE = logical 0.

➤ Controlling devices running at a higher voltage than the one on your prototyping platform requires using relays to interface the logic with whatever you want to control.

➤ The different digital sensors can use the same code. For example, a tilt sensor is read the same way a pushbutton is read. The difference between them has to do with their physical affordances, thus the way the user will interact with them, the way they will hold it in their hands, or how it will react to different movements, presses, etc.

In diploma work, we developed communication system based on the Arduino. For communication line, I use digital input/output ports which send/receive digital signals. Designing and developing of digital communication systemsare defined and source code is given.

LITERATURE LIST

1. Abhay Maheshwari, Soon-Shin Chee, "Board Routability Guidelines with Xilinx Fine-Pitch BGA Packages", Xilinx Application Notes, XAPP157 (v1.2), November 2012. pp. 74-112.

2. Amp Inc, "MICTOR SB Connector", Product information, Engineering Test Report: CTLB042234-005, June 2013. pp. 36-54.

3. Austin Lesea and Mark Alexander, "Powering Xilinx FPGAs", Xilinx Application Notes, XAPP158 (v1.5), August 2010. pp. 25-46.

4. Alteon Networks, "Tigon/PCI Ethernet Controller", Revision 1.04, August 2009. pp. 18-39.

5. Alteon WebSystems, "Gigabit Ethernet/PCI Network Interface Card: Host/NIC Software Interface Definition", Revision 12.4.13, July 2010. pp. 22-46.

6. Bruce Oakley, Bruce Brown, "Highly Configurable Network Interface Solutions on a Standard Platform", AMIRIX Systems Inc., 2012. pp. 72-88.

7. Boden, Nan, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su "Myrinet, a Gigabit per Second Local Area Network", IEEE-Micro, Vol.15, No.1, pp. 29-36, February 2008.

8. Carol Fields, "Design Reuse Strategy for FPGAs", Xilinx Design Solutions Group, XCell 37, 2013. pp. 132-146.

9. Charles E. Spurgeon. "Ethernet The Definitive Guide" O'Reilly & Associates, Inc.2010. pp. 122-136.

10. F. Petrini, S. Coll, E. Frachtenberg, and A. Hoisie. "Hardware- and Software-Based Collective Communication on the Quadrics Network". In

IEEE International Symposium on Network Computing and Applications 2001, (NCA 2001), Boston, MA, February 2012. pp. 146-167.

11. Fabrizio Petrini, Adolfy Hoisie, Wu chun Feng, and Richard Graham. "Performance Evaluation of the Quadrics Interconnection Network". In Workshop on Communication Architecture for Clusters (CAC '01), San Francisco, CA, April 2011. pp. 188-202.

12. Field Programmable Port Extender Homepage, Available from:http://www.arl.wustl.edu/projects/fpx , Aug.2014.

13. Model Sim Inc., "Modelsim Xilinx User's Manual.", Available from: http://www.model.com.

14. Myricom Inc., "Myrinet Software and Customer Support", Available from: http://www.myri.com/scs/GM/doc , 2014.

15. Xilinx Inc, "MicroBlaze Reference Guide. EDK (v3.2)", Available from: http://www.xilinx.com/ise/embedded/edk_docs.htm , April 2013.

16. Xilinx Inc, " FPGA Xpower tutorial". Available from: http://toolbox.xilinx.com/docsan/xilinx5/help/xpower/xpower.htm.

17. Xilinx Inc, "PowerPC Embedded Processor Solution" available from: http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=v2p_powerpc

18. Экология и безопасность жизнедеятельности: Учебное пособие для студентов ВУЗов/ ред. Л. А. Муравий, 2008. 34-79 л.

19. Белов С.В. Безопасность жизнедеятельности М.: Высшая школа. 2009.