EasyChair Preprint

№ 5474

Dam Level Warning Using GSM

Gundam Rahul, Bojedla Rasi and Mudalkur Nithesh

May 8, 2021

CHAPTER 1

INTRODUCTION

Dams are the major sources of water supply to cities, they also play a vital role in flood control and can assist river navigation. Most of the dams are built to serve more than one purpose and their benefits are manifold. It is necessary to implement some sort of communication between the metering systems and computer models to provide support in managing the complex systems of the hydro power plants. Generally, the dams are monitored through traditional surveillance techniques and the water management except the monitoring of level of water in some of the dams which is automatized. Management of water resources through dams becomes complex as the number of users depending on dams is huge and these users may have conflicting interests. This situation gets much complex with the fact that the available resources are limited with high possibilities of droughts and floods. This affects the densely populated areas. Dam monitoring is a tedious and long term process which has to be improved step by step. A new system for dam water monitoring and management should be established which can provide water level in real time and can allow us to come to quick conclusions regarding the safety operations of the dams. Internet of Things (IoT) can be defined as a network of devices which are interconnected. It comprises a set of sensors, communication network as well as software enabled electronic devices that enables end users to acquire accurate data from time to time through the communication channel and allows for data interchange between users and the connected devices.

This system can be used to automatize the control of dams without human interference. This can also be used to gather information on the level of water throughout the country and can be used to route water based on the requirements. We can get information on the water availability in a particular region and route the water to that area if there's scarcity. This helps a lot in irrigation. Keeping a check on the safety of dam from time to time is one of the important measure to ensure the safety of dams. Use of Wireless sensors network with software for dam safety management helps in improving the functionality of dams. All the sensors in the cluster of dam such as Water Level Sensor ,Vibration Sensor and Pressure Sensor can be used to sense Water level Vibrations on the wall of dam and Pressure exerted on the wall of dam from the dam into the main pipeline in Litres per minute respectively.Differential Pressure sensors are fitted at equal spaces along the main pipeline

which can sense the pressure difference because of the breaking or leakage of the pipeline and will immediately be communicated to the observer. In case of floods the routing of flood water can be done more efficiently considering the level of water across different dams. Surveillance of areas near the dams can be done using cameras which transmit live footage to the base station and will be helpful in identifying the presence of people near the dams and can help in ensuring safety while releasing water during flash floods. Internet of Things technology focusses on making the ecosystem of sensors more and more intelligent by establishing a connection to the internet. Collecting the data regarding the failed sensors enables us to generate more reliable equipment which in turn improves the reliability of the dams. Integration of Internet of Things with big data, cloud computing and WSN will enhance the operation capability to dams to a greater extent. The entire processing of data will be done on the cloud which will ensure that the data retrieval and issuing of commands can be made faster with more reliability.

1.1 OBJECTIVE OF THE PROJECT

The aim of this project is design and develop a Dam water level monitoring system using GSM, which is simple and low cost device.

1.2 PROPOSED SYSTEM

A prototype of the proposed idea has been implemented using GSM communication,Water level sensors and Arduino micro controller. The first stage of the implementation which involves determining the level of water using water level sensors. The water level sensor is mounted on the top of a DAM which determines the distance between the top of the Dam and the surface of the water. If the distance goes below a certain point it indicates that the water level in the dam has changed its level. As soon as the level of water changes the micro controller will send masseage to the authorized person.

a)BLOCK DIAGRAM

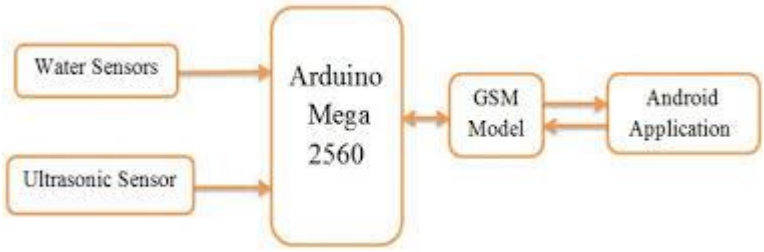


Fig 1.1: Block diagram

b) MAJOR BUILDING BLOCKS OF THIS PROJECT

1. Arduino Microcontroller
2. Water Level sensor(Ultra Sonic Sensor) s
3. GSM Module

c) SCHEMATIC DIAGRAM

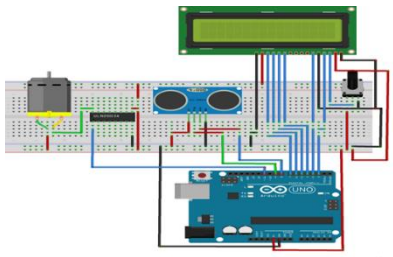


Fig 1.2: Schematic Diagram

CHAPTER 2

LITERATURE SURVEY

Level controlling of water or liquid is a very active field where many papers have addressed the fuzzy or neural networks control in the water level control system. It requires measurement and control of the water-level.

One of the easiest way to measure water level is using submersible pressure transducers (wet sensors) which are easy to install and require very little maintenance. For the very reason, they are often used for temporary installations and installations in remote locations. They are supposed to be installed in a fixed position and should remain fully submersed at all times. It works on the concept of application of hydrostatic pressure to a strain gauge, which converts mechanical movement into an electrical signal which is in turn measured by the station data logger and converted into pressure, level and discharge.

As we see, now a days the safety control of large dams depends mainly on the measurement of some important quantities like absolute and relative displacements, strains and stresses in the concrete, discharges through the foundation, etc. and on visual inspections of the dam structures. In certain dams, the analysis of the measured data is compared with results of mathematical or physical models and is helpful in the structural safety assessment.

Artificial neural networks (ANNs) provide a quick and flexible means of creating models for river flow prediction, and have been shown to perform well in comparison with conventional methods. These networks can be used for characterising the normal structural behaviour of the dams by taking into account the actions to which the structure is subject to in the past. An artificial neural network is nothing but an interconnected group of nodes, similar to the vast network of neurons in a brain. Each circular node of this network represents an artificial network and an arrow represents flow from the output of one node to the input of another.

A dam in its lifetime, can be exposed to significant water level variations and seasonal environmental temperature changes. The most important factor in water supplying is to supply the required water quantity while maintaining a water head above a minimum limit. It is required to deliver the necessary quantity of water with the adequate pressure to the final consumers. The overhead storage tanks are built so that the water level could be kept at a constant value to maintain that pressure. The demand of the consumers is changing throughout the day and that leads to change the output flow rate of the tank. If water is

supplied at a constant rate either over flow of water or pressure drops at end user may happen. This leads to the necessity of water level management.

One such project was done on river Nile by establishing national geo referential databases and spatial layers along with hydro-meteorological parameters, water usage information, land use, extent of land cover and soil types. Present day monitoring technologies such as automatized weather stations and acoustic based Doppler current profilers were introduced. This sort of equipment is coherent for water flow measurement. In majority of the rivers the Doppler current profiler is mounted on a boat and operated from there. This scraps the need for the construction of an expensive cableway. This project also constitutes two buoys to measure the extent of water evaporation on Lake Nasser in Egypt, serving as a means to check the ground truth for the satellite based evaporation computations. Most of the basin countries have adopted this data structure, which was designed in an interactive process with the respective water resource agencies. This generated data structure ensures seamless and rapid data exchange once it has set up proper mechanisms for data sharing. A common structure is also necessary for developing basin-wide assessment instruments like the Nile DST.

However, the South-East river trust employed a method called weir for water level control in river Teise .A weir is basically a barrier built across a river to alter the flow characteristics. In most cases weirs are constructed in the form of a horizontal barrier across the width of a river that pools water behind it whilst still allowing it to flow steadily over the top. Weirs are often used to prevent flooding and measure discharge.

Another monitoring system was developed for measurement of water levels, and it is composed of ultrasonic sensor, PIC micro-controller, and GSM module. The ultrasonic sensor measures the distance from the sensor to the liquid surface. This system proposes the development of water level monitoring system by integrating the GSM module to alert the person-in-charge through Short Message Service (SMS) when the water has reached the critical level and it will automatically turn OFF the pump. It is possible to monitor the level of water whenever required.

In China, computing techniques are deployed to curb wastage of water and generate better financial gains. It also ensures the conservation of environment and water cycle so that we can pass on the water resources to our future generations. They used systems constituting Arduino for the automation of pumping of water into the tanks with the help of sensors which can sense the level of water in the tank. The pump system will function

automatically based on the level of water and a LED screen is used to keep the user informed regarding the status. This system can also be extended to automate the extraction of water from sump tank. The same strategy for measuring the level of water using sensors in the sump can be used nut here if the system senses that the water level is low it prevents the motor from running to ensure safety from dry running. An alert sound can be initiated in that case to alert the user regarding the issue.

The current dam control technology is manual wherein the handler operates the gate on command. This gives room for irregular water sharing between two properties, human error which can result in floods or unnecessary wastage of water. Our proposed system removes these possibilities incorporating automated dam control system. The entire system is also partially self-powered. The design involves energy harvesting thereby making the system self-sustained.

CHAPTER 3

INTRODUCTION TO EMBEDDED SYSTEMS

3.1 GENERAL INTRODUCTION

An embedded system is a combination of software and hardware to perform a dedicated task. Some of the main devices used in embedded products are Microprocessors and Microcontrollers. Microprocessors are commonly referred to as general purpose processors as they simply accept the inputs, process it and give the output.

In contrast, a microcontroller not only accepts the data as inputs but also manipulates it, interfaces the data with various devices, controls the data and thus finally gives the result. In this project we use RF module as well as the DTMF decoder for communication. Now when we dial the numbers in the mobile phone from the controlling side then it automatically recognizes which number has been recorded and it follows with the corresponding next step to be taken i.e., movement of the robot in water.

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. A good example is the microwave oven. Almost every household has one, and tens of millions of them are used every day, but very few people realize that a processor and software are involved in the preparation of their lunch or dinner.

This is in direct contrast to the personal computer in the family room. It too is comprised of computer hardware and software and mechanical components (disk drives, for example). However, a personal computer is not designed to perform a specific function rather; it is able to do many different things. Many people use the term general-purpose computer to make this distinction clear. As shipped, a general-purpose computer is a blank slate; the manufacturer does not know what the customer will do wish it. One customer may use it for a network file server another may use it exclusively for playing games, and a third may use it to write the next great American novel.

3.2 Block Diagram of Embedded System

Now, the details of the various building blocks of the hardware of an embedded system as shown in fig 3.1 are

Central processing unit (CPU)
Memory (Read-only Memory and Random Access Memory)
Input Devices

Output Devices

Communication interfaces

Applications-specific circuitry

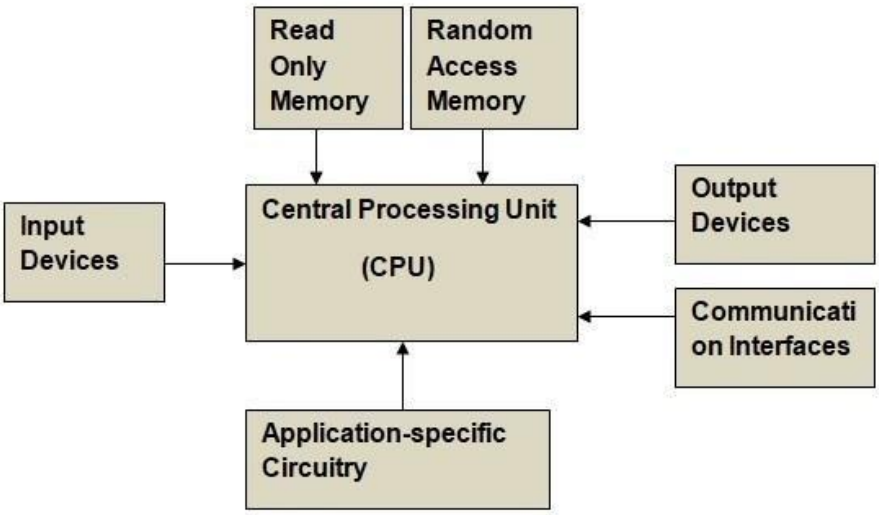


Fig 3.1: Block Diagram Of Embedded Systems

3.2.1 Central Processing Unit (CPU)

The Central Processing Unit (processor, in short) can be any of the following microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low- cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog-to digital converter etc. so, for small applications, a micro-controller is the best choice as the number of external component required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. DSP is used for many applications for signal processing.

3.2.2 Memory

The memory is categorized as Random Access Memory (RAM) and Read Only Memory (ROM). The contents of the RAM Will be erased if power is switched off of to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; the program is executed.

3.2.3 Input devices

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is not an easy task. Many embedded systems will have a small keyboard-you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device for user interaction; they take inputs from sensors are transducers and produce electrical signals that are in turn fed to other systems.

3.2.4 Output devices

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a few Light Emitting Diodes (LEDs) to indicate the health status of the system modules, or for visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display some important parameters.

3.2.5 Communication interfaces

The embedded system may need to, interact with other embedded system at they may have to transmit data to a desktop. To facilitate this, the embedded system are provided with one or a few communication interfaces such as RS232, RS422,RS485, Universal Serial Bus (USB), and IEEE 1394, Ethernet etc.

3.2.6 Application-specific circuitry

Sensors, transducers, special processing and control circuitry may be required for an embedded system, depending on its application. This circuitry interacts with the processor carry out the necessary work. The enter hardware has to be given power supply either through the 230 volts main supply or through a battery. The hardware as to design in such a way that the power consumption is minimized. Security is the condition being protected against danger or loss. In the general sense, security is a concept similar to safety. The nuance between the two is an added emphasis on being protected from dangers that originate from outside. Individuals or actions that encroach up on the condition of protection or responsible for the breach of security. The word “security” in general usage is synonymous with “safety,” but as a technical “security” means that something not only is secure but that it has be secured. One of the best options for providing good security is by using a technology named EMEDDED SYSTEM.

3.3 OVERVIEW OF EMBEDDED SYSTEMS

An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

Embedded systems are controlled by one or more main processing cores that are typically either microcontrollers or digital signal processors (DSP). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may usefully be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites. (Each radar probably includes one or more embedded systems of its own.)

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

In general, "embedded system" is not a strictly definable term, as most systems have some element of extensibility or programmability. For example, handheld computers share some elements with embedded systems such as the operating systems and microprocessors which power them, but they allow different applications to be loaded and peripherals to be connected. Moreover, even systems which don't expose programmability as a primary feature generally need to support software updates. On a continuum from "general purpose" to "embedded", large application systems will have subcomponents at most points even if the system as a whole is "designed to perform one or a few dedicated functions", and is thus

appropriate to call "embedded". A modern example of embedded system is shown in fig: 3.2.

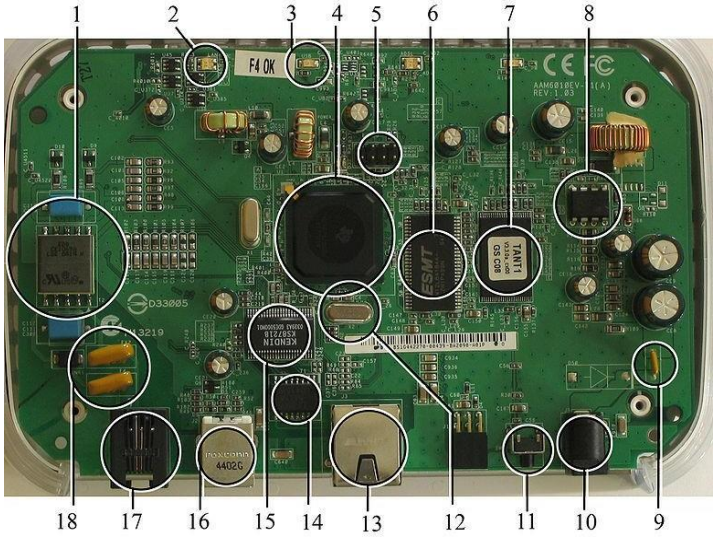


Fig 3.2: A Modern Example Of Embedded Systems

Labeled parts include microprocessor (4), RAM (6), flash memory (7).Embedded systems programming is not like normal PC programming. In many ways, programming for an embedded system is like programming PC 15 years ago. The hardware for the system is usually chosen to make the device as cheap as possible. Spending an extra dollar a unit in order to make things easier to program can cost millions. Hiring a programmer for an extra month is cheap in comparison. This means the programmer must make do with slow processors and low memory, while at the same time battling a need for efficiency not seen in most PC applications. Below is a list of issues specific to the embedded field.

3.4 HISTORY

In the earliest years of computers in the 1930–40s, computers were sometimes dedicated to a single task, but were far too large and expensive for most kinds of tasks performed by embedded computers of today. Over time however, the concept of programmable controllers evolved from traditional electromechanical sequencers, via solid state devices, to the use of computer technology.

One of the first recognizably modern embedded systems was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Automatics

D-17 guidance computer for the Minuteman missile, released in 1961. It was built from transistor logic and had a hard disk for main memory. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits.

3.4.1 Tools

Embedded development makes up a small fraction of total programming. There's also a large number of embedded architectures, unlike the PC world where 1 instruction set rules, and the UNIX world where there's only 3 or 4 major ones. This means that the tools are more expensive. It also means that they're lowering featured, and less developed. On a major embedded project, at some point you will almost always find a compiler bug of some sort.

Debugging tools are another issue. Since you can't always run general programs on your embedded processor, you can't always run a debugger on it. This makes fixing your program difficult. Special hardware such as JTAG ports can overcome this issue in part. However, if you stop on a breakpoint when your system is controlling real world hardware (such as a motor), permanent equipment damage can occur. As a result, people doing embedded programming quickly become masters at using serial IO channels and error message style debugging.

3.4.2 Resources

To save costs, embedded systems frequently have the cheapest processors that can do the job. This means your programs need to be written as efficiently as possible. When dealing with large data sets, issues like memory cache misses that never matter in PC programming can hurt you. Luckily, this won't happen too often- use reasonably efficient algorithms to start, and optimize only when necessary. Of course, normal profilers won't work well, due to the same reason debuggers don't work well.Memory is also an issue. For the same cost savings reasons, embedded systems usually have the least memory they cZan get away with. That means their algorithms must be memory efficient (unlike in PC programs, you will frequently sacrifice processor time for memory, rather than the reverse). It also means you can't afford to leak memory It also means you can't afford to leak memory. Embedded applications generally use deterministic memory techniques and avoid the default "new" and "malloc" functions, so that leaks can be found and eliminated more easily. Other resources programmers expect may not even exist. For example, most embedded processors do not have hardware FPUs (Floating-Point Processing Unit). These resources either need to be emulated in software, or avoided altogether.

3.4.3 Real Time Issues

Embedded systems frequently control hardware, and must be able to respond to them in real time. Failure to do so could cause inaccuracy in measurements, or even damage hardware such as motors. This is made even more difficult by the lack of resources available. Almost all embedded systems need to be able to prioritize some tasks over others, and to be able to put off/skip low priority tasks such as UI in favor of high priority tasks like hardware control.

3.5 NEED FOR AN EMBEDDED SYSTEMS

The uses of embedded systems are virtually limitless, because every day new products are introduced to the market that utilizes embedded computers in novel ways. In recent years, hardware such as microprocessors, microcontrollers, and FPGA chips have become much cheaper. So when implementing a new form of control, it's wiser to just buy the generic chip and write your own custom software for it. Producing a custom-made chip to handle a particular task or set of tasks costs far more time and money. Many embedded computers even come with extensive libraries, so that "writing your own software" becomes a very trivial task indeed. From an implementation viewpoint, there is a major difference between a computer and an embedded system. Embedded systems are often required to provide Real-Time response. The main elements that make embedded systems unique are its reliability and ease in debugging.

3.5.1 Debugging

Embedded debugging may be performed at different levels, depending on the facilities available. From simplest to most sophisticate they can be roughly grouped into the following areas:

- Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic).
- External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multi core systems.
- An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface. This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.

- An in-circuit emulator replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.
- A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified and allowing debugging on a normal PC.

Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation. The view of the code may be as assembly code or source-code.

Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary. For instance, debugging a software (and microprocessor) centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals (DSP, FPGA, co-processor). An increasing number of embedded systems today use more than one single processor core. A common problem with multi-core development is the proper synchronization of software execution. In such a case, the embedded system design may wish to check the data traffic on the busses between the processor cores, which requires very low-level debugging, at signal/bus level, with a logic analyzer, for instance.

3.5.2 Reliability

Embedded systems often reside in machines that are expected to run continuously for years without errors and in some cases recover by them if an error occurs. Therefore the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided.

Specific reliability issues may include:

- The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
- The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals, engines on single-engine aircraft.
- The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated

sales and service.

A variety of techniques are used, sometimes in combination, to recover from errors both software bugs such as memory leaks, and also soft errors in the hardware:

- Watchdog timer that resets the computer unless the software periodically notifies the watchdog
- Subsystems with redundant spares that can be switched over to
- software "limp modes" that provide partial function
- Designing with a Trusted Computing Base (TCB) architecture[6] ensures a highly secure & reliable system environment

An Embedded Hypervisor is able to provide secure encapsulation for any subsystem component, so that a compromised software component cannot interfere with other subsystems, or privileged-level system software. This encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection.

- Immunity Aware Programming

3.6 EMBEDDED SOFTWARE

Various software's can be for various purposes. ALP i.e. Assembly Language program can be used as the back end the other software like C, VB, etc. can be in front end.

3.7 EMBEDDED SOFTWARE ARCHITECTURE

There are several different types of software architecture in common use.

3.7.1 Simple Control Loop

In this design, the software has a loop. The loop calls the subroutines, each of which manages a part of the hardware or software.

3.7.2 Interrupt Controlled System

Some embedded systems are predominantly interrupt controlled. This means that tasks performed by the system are triggered by different kinds of events. An interrupt could be generated for example by a timer in a pre-defined frequency, or by a serial port controller receiving a byte.

These kinds of the systems run a simple task in a main loop also, but this task is not sensitive to expected delays. The tasks performed in the interrupt handlers should be kept short the interrupt latency to a minimum. Some times longer tasks are added to a queue structure in the interrupt handlers to be processed in the main loop later.

3.7.3 Pheripherals

Peripherals are the various devices that are connected to the CPU, for performing various functions. Embedded systems talk with the outside world via peripherals, such as

Serial communication interfaces (SCI) RS-232, RS-422, RS-458 etc.

Synchronous serial communication interfaces (SSCI) 12C, JTAG, SPI, SSC and ESSI Universal serial bus (USB)

Network Controller area Network, etc.

Timer PLL(s), Capture/Compare and processing units.

Discrete I/O General Purpose Input/output(GPIO).

3.7.4 Processors

Processors are the key elements in any embedded system. They interact with the memory, where the various instructions of useful functions into a single IC package.

The ability to execute a stored set of instructions to carry out user defined tasks .The ability to be able to access external memory chips to both read and writes data from and to the memory.

3.7.5 Micro Controller

Basically, a microcontroller is a device which integrates a number of the components of a microprocessor system on to a single chip. So a microcontroller combines onto the same microchip CPU core Memory (both ROM and RAM) Most microcontrollers will also combine other devices such as

A Timer module to allow the microcontroller to perform tasks for certain periods. A serial I/O port to allow data to flow between the microcontroller and other devices such as a PC or another microcontroller. An ADC to allow the microcontroller to accept analogue input data for processing.

3.7.6 DSP

It is the study of the signals in digital representation and processing methods of these signals. It is used where large mathematical and scientific calculations are required.

3.7.7 ASIC

It is an IC designed for a specific application. This IC designed for specific application can't be used for other application.

3.8 EXPLANATION OF EMBEDDED SYSTEMS

3.8.1 Software Architecture

There are several different types of software architecture in common use.

- **Simple Control Loop:**

In this design, the software simply has a loop. The loop calls subroutines, each of which manages a part of the hardware or software.

- **Interrupt Controlled System:**

Some embedded systems are predominantly interrupt controlled. This means that tasks performed by the system are triggered by different kinds of events. An interrupt could be generated for example by a timer in a predefined frequency, or by a serial port controller receiving a byte. These kinds of systems are used if event handlers need low latency and the event handlers are short and simple.

Usually these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays. Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

- **Cooperative Multitasking**:

A non-preemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to "run" in. When a task is idle, it calls an idle routine, usually called "pause", "wait", "yield", "nop" (stands for no operation), etc. The advantages and disadvantages are very similar to the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue-interpreter.

- **Primitive Multitasking:**

In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

As any code can potentially damage the data of another task (except in larger systems using an MMU) programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues, semaphores or a non- blocking synchronization scheme.

Because of these complexities, it is common for organizations to buy a real-time operating system, allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a generic real time system, due to limitations regarding memory size, performance, and/or battery life.

Microkernels and Exokernels:

A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when the task switching and intertask communication is fast, and fail when they are slow. Exokernels communicate efficiently by normal subroutine calls. The hardware and all the software in the system are available to, and extensible by application programmers. Based on performance, functionality, requirement the embedded systems are divided into three categories.

3.8.2 Standard Alone Embedded systems

These systems takes the input in the form of electrical signals from transducers or commands from human beings such as pressing of a button etc.., process them and produces desired output. This entire process of taking input, processing it and giving output is done in standalone mode. Such embedded systems comes under stand-alone embedded systems.

Eg: microwave oven, air conditioner etc..

3.8.3 Real Time Embedded Systems

Embedded systems which are used to perform a specific task or operation in a specific time period those systems are called as real-time embedded systems. There are two types of real- time embedded systems.

- **Hard Real-time embedded systems:**

These embedded systems follow an absolute dead line time period i.e.., if the tasking is not done in a particular time period then there is a cause of damage to the entire equipment.

Eg: consider a system in which we have to open a valve within 30 milliseconds. If this valve is not opened in 30 ms this may cause damage to the entire equipment. So in such cases we use embedded systems for doing automatic operations.

- **Soft Real Time embedded systems:**

Eg: Consider a TV remote control system, if the remote control takes a few milliseconds delay it will not cause damage either to the TV or to the remote control. These systems which will not cause damage when they are not operated at considerable time period those systems comes under soft real-time embedded systems.

3.8.4 Network Communication Embedded Systems

A wide range network interfacing communication is provided by using embedded systems.

- Consider a web camera that is connected to the computer with internet can be used to spread communication like sending pictures, images, videos etc.., to another computer with internet connection throughout anywhere in the world.
- Consider a web camera that is connected at the door lock.

  Whenever a person comes near the door, it captures the image of a person and sends to the desktop of your computer which is connected to internet. This gives an alerting message with image on to the desktop of your computer, and then you can open the door lock just by clicking the mouse. Fig: 2.2 show the network communications in embedded systems.

Fig 3.3: Network Communication Embedded Systems

3.8.5 Different Types Of Processing Units

The central processing unit (c.p.u) can be any one of the following microprocessor, microcontroller, digital signal processing.

- Among these Microcontroller is of low cost processor and one of the main advantage of microcontrollers is, the components such as memory, serial communication interfaces, analog to digital converters etc.., all these are built on a single chip. The numbers of external components that are connected to it are very less according to the application.

- Microprocessors are more powerful than microcontrollers. They are used in major applications with a number of tasking requirements. But the microprocessor requires many external components like memory, serial communication, hard disk, input output ports etc.., so the power consumption is also very high when compared to microcontrollers.

- Digital signal processing is used mainly for the applications that particularly involved with processing of signals.

3.9 FEATURES

The main feature of an embedded systems are its reliability and the scope for debugging.

3.9.1 Debugging

Debugging may be performed at different levels, depending on the facilities available, ranging from assembly or source level debugging with an in circuit emulator or in circuit debugger, to outputs from serial debug ports to an emulated environment running on a PC. As the complexity of embedded systems grows, higher level tools and operating

systems or migrating into machinery where it makes sense.

3.9.2 Reliability

Embedded systems often reside in machines that are expected to run continuously for years without errors and in some cases recover by themselves if any error occurs. Therefore the software is usually developed and tested more carefully than that for PC, and unreliable mechanical moving parts such as disk drivers, switches or buttons are avoided.

Specific reliable issues may include the system cannot safely be shut down for repair, or it is too inaccessible to repair. Solutions may involve subsystems with redundant spares that can be switched over to, or software "limp modes" that provide partial function. Examples include space systems, undersea cables, navigational beacons, bore-hole systems and automobiles. The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft, navigation, reactor control systems, safely critical chemical factory controls, train signals and engines on single-engine aircraft.

3.10 APPLICATIONS OF EMBEDDED SYSTEMS

3.10.1 Consumer Applications

At home we use a number of embedded systems which include microwave oven, remote control, vcd players, DVD players, camera etc….

3.10.2 Office Automation

We use systems like fax machine, modem, printer etc…

Fig 3.4: Fax Machine

3.10.3 Industrial Automation

Today a lot of industries are using embedded systems for process control. In industries we design the embedded systems to perform a specific operation like monitoring temperature, pressure, humidity ,voltage, current etc.., and basing on these monitored In critical industries where human presence is avoided there we can use robot**s** which are programmed to do a specific operation.

Fig 3.5: Robot

3.10.4 Computer Networking

The most efficient types of the network used in the embedded systems are BUS networkand an Ethernet network. A BUS is used to connect different network devices and to transfer a huge range of data, for example, serial bus, I2C bus, CAN bus, etc. The Ethernet type network works with the TCP/IP protocol

Fig 3.6: Computer Networking

CHAPTER 4

HARDWARE DESCRIPTION

4.1 ARDUINO

Fig 4.1: Arduino Board

The Arduino uno R3 is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground ,making it easier to DFU mode.

- **1.0 pinout:-**

Added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.

- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno

is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
InputVoltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power:

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN:** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V:**The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3:** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND:** Ground pins.

Memory:

The ATmega328 has 32 KB (with 0.5 KB used for the boot loader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output:

Each of the 14 digital pins on the Uno can be used as an input or output, using pin Mode(), digital Write(), and digital Read()functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 k Ohms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX):** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach Interrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11:**Provide 8-bit PWM output with the analog Write() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK):** These pins support SPI communication using the SPI library.

- **LED: 13:**There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference() function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin:** Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- **AREF:** Reference voltage for the analog inputs. Used with analog Reference().
- **Reset:** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming:

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the tools>Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU boot loader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU boot loader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of theATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarade capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the boot loader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the boot loader is running on the Uno. While

it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB over current Protection:

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and over current. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

4.2 OTHER COMPONENTS USED IN THIS PROJECT

a)Ultrasonic Sensor

Fig 4.2: Ultrasonic Sensor

This HC-SR04-Ultrasonic Range Finder is a very popular sensor which is found in many applications where it requires to measure distance and detect the objects.

The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The HC-SR04 ultrasonic sensor uses sonar to determine the distance to an object like bats or dolphins do.

This Ultrasonic Sensor module is a transmitter, a receiver and a control circuit in one single pack!! It has very handy and compact construction. It offers excellent range accuracy and

stable readings in an easy-to-use package. Its operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect).

The Trigger and the Echo pins are the I/O pins of this module and hence they can be connected to the I/O pins of the microcontroller. When the receiver detects return wave the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor.

Ultrasonic Ranging Module HC-SR04 provides 2cm-400cm non-contact distance sensing capabilities, Ranging accuracy up to 3mm.

This Ultrasonic Sensor can be attached to your project using mounting bracket, so buy it now at Robu.in we have a very good quality Acrylic mounting bracket for this HC-SR04 Ultrasonic Module.

Wiring
+5V(positive)
Trig(control)
Echo(receive)
GND(negative)

Features:

1. Measures the distance within a wide range of 2cm to 400cm
2. Stable performance
3. Accurate distance measurement
4. High-density

b)Gsm Module

Fig 4.2: Gsm Module

SIM808 GSM/GPRS/GPS UART Mini Modem, is a high quality commercial grade product from rhydoLABZ, which is professionally designed with impedance matching RF PCB designs and is built with Quad Band GSM/GPRS engine-SIM808 from SIMCOM, works on frequencies 850/ 900/ 1800/ 1900 MHz which combines GPS technology for satellite navigation. It is designed as very compact to be used as tracker. The GSM/GPRS/GPS Modem is having internal TCP/IP stack to enable you to connect with internet via GPRS. It is suitable for Tracking, SMS, Voice as well as DATA transfer application in M2M interface and vehicle tracking applications. This compact modem does not have onboard voltage regulator IC, it should be used with 3.7V Li-Po battery, which will charge automaticaly using onboard charger while connecting with 5V DC Power via header pins.

SIM808 GSM/GPRS/GPS UART Mini Modem**,** is a high quality commercial grade product from rhydoLABZ, which is professionally designed with impedance matching RF PCB designs and is built with Quad Band GSM/GPRS engine-SIM808 from SIMCOM, works on frequencies 850/ 900/ 1800/ 1900 MHz which combines GPS technology for satellite navigation. It is designed as very compact to be used as tracker. The GSM/GPRS/GPS Modem is having internal TCP/IP stack to enable you to connect with internet via GPRS. It is suitable for Tracking, SMS, Voice as well as DATA transfer application in M2M interface and vehicle tracking applications. This compact modem does not have onboard voltage regulator IC, it should be used with 3.7V Li-Po battery, which will charge automaticaly using onboard charger while connecting with 5V DC switch has been provided to ON/OFF manually.

Sliding type Micro SIM Socket has been provided on bottom side of the board to reduce the PCB size and the SIM Pins are protected from antistatic high voltages using onboard Transient Suppressors. All audio interface connections are available on Headphone connector after necessary filter circuits. The Modem is populated with Edge mountable SMA Connector for GSM Antenna and UFL connector for external GPS Antennas. USB connector has been provided for updating the module Firmware if needed, it also serves as battery charger connector. 3LED's are provided to indicate the modem status. Most of the Serial port interfaces are made available at Bergstrip connector along with Audio interface connections, which can be used if needed in your circuits.The Modem is manufactured with Automatic Pick and place machine with high quality standard. LiPo battery is used to power up the module. The baud rate is configurable from 9600-115200 through AT command.

Features of SIM808 GSM/GPRS/GPS Mini Modem

- High Quality Product(Not hobby grade)
- Professionally Designed Modem with Impedance Maching Layout.
- Manufactured by Fully Automatic Pick and Place Machine.
- Onboard SMA connector for GSM antenna
- Onboard UFL connector to Connect Patch antenna for GPS
- Onboard Micro USB Connector for module firmware updation
- Automatic and Manual Start option (SMD Switch has provided for Manual Startup)
- Onboard Connector for connecting 3.7V Li-Po battery
- Option for battery charging
- onboard Headphone jack for audio interface
- Protection to prevent Accidental over voltage
- onboard TXB0104 voltage level translator IC which helps us to interface with 2.8V to 5V MicroControllers
- onboard LED for power indication, Network and status.
- Onboard SIM Socket Anti Static Protection TVS
- Onboard Micro sim card connector
- Two Mounting Holes are Provided
- Easy to Make and receive voice calls, Send and receive SMS messages
- Input voltage : 3.4 to 4.2 Dc or 3.7V Li-Po Battery...
- Operation temperature:-40°C ~85°C

Specifications of SIM808 Module

- Quad-band 850/900/1800/1900MHz
- GPRS multi-slot class 12/10
- GPRS mobile station class B
- Compliant to GSM phase 2/2+ – Class 4 (2 W @ 850/900MHz) – Class 1 (1 W @ 1800/1900MHz)
- Dimensions: 24*24*2.6mm
- Control via AT commands (3GPP TS 27.007, 27.005 and SIMCOM enhanced AT Commands)
- Supply voltage range 3.4 ~ 4.4V
- Low power consumption

- Operation temperature:-40°C ~85°C
- Power supply : 3.4V ~ 4.4V
- Charging : Supports charging control for Li-Ion battery

Specifications for GPRS Data

- GPRS class 12: max. 85.6 kbps (downlink/uplink)
- PBCCH support
- Coding schemes CS 1, 2, 3, 4
- PPP-stack
- USSD

Specifications for SMS via GSM/GPRS

- Point to point MO and MT
- SMS cell broadcast
- Text and PDU mode

Specification for GPS

- Receiver type
  - 22 tracking /66 acquisition
  - channel-GPS L1 C/A code
- Sensitivity
  - Tracking: -165 dBm
  - Cold starts : -148 dBm
- Time-To-First-Fix
  - Cold starts: 32s (typ.)
  - Hot starts: <1s
  - Warm starts: 3s

Accuracy-Horizontal position : <2.5m CEP

CHAPTER 5

SOFTWARE DEVELOPMENT

5.1 INTRODUCTION TO IDE

(Integrated Development Environment)

This tutorial will walk you through downloading, installing, and testing the Arduino software (also known as the Arduino IDE - short for Integrated Development Environment). Before you jump to the page for your operating system, make sure you've got all the right equipment.

Requirements

- A computer (Windows, Mac, or Linux).
- An Arduino-compatible microcontroller (anything from this guide should work).
- A USB A-to-B cable, or another appropriate way to connect your Arduino-compatible microcontroller to your computer (check out this USB buying guide if you're not sure which cable to get).

Fig 5.1: An A-to-B USB Cable

5.1.1 Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of

instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IOT applications, wearable, 3D printing, and embedded environments

.

Fig 5.2: Arduino

All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics.

Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

5.1.2 Installing Arduino in Different Operating Systems

This page will show you how to install and test the Arduino software with a Windows operating system (Windows 8, Windows 7, Vista, and XP).

Windows 8, 7, Vista, and XP

- Go to the Arduino download page and download the latest version of the Arduino software for Windows.
- When the download is finished, un-zip it and open up the Arduino folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is important so don't be moving any files around unless you really know what you're doing.
- Power up your Arduino by connecting your Arduino board to your computer with a USB cable (or FTDI connector if you're using an Arduino pro).
- You should see the LED labelled 'ON' light up. (This diagram shows the placement of the power LED on the UNO).
- If you're running Windows 8, you'll need to disable driver signing, so go see the Windows 8 section. If you're running Windows 7, Vista, or XP, you'll need to install some drivers, so head to the Windows 7, Vista, and XP section down below.

Windows 8

Windows 8 comes with a nice little security 'feature' that 'protects' you from unsigned driver installation. Some older versions of Arduino Uno come with unsigned drivers, so in order to use your Uno, you'll have to tell Windows to disable driver signing. This issue has been addressed in newer releases of the Arduino IDE, but if you run into issues, you can try this fix first. For a nice, step-by-step tutorial with pictures click here, otherwise the steps are outlined below.

To Temporarily Disable Driver Signing:

- From the Metro Start Screen, open Settings (move your mouse to the bottom-right-corner screen and wait for the pop-out bar to appear, then click the Gear icon)
- Click 'More PC Settings'
- Click 'General'

- Scroll down, and click ‘Restart now’ under ‘advanced startup’.
- Wait a bit.
- Click ‘Troubleshoot’.
- Click ‘Advanced Options’
- Click ‘Windows Startup Settings’
- Click Restart.
- When your computer restarts, select ‘Disable driver signature enforcement ‘from the list.

To permanently disable driver signing (recommended, but has some minor security implications):

- Go to the metro start screen
- Type in “cmd”
- Right click “Command Prompt” and select “Run as Administrator” from the buttons on the bottom of your screen
- Type paste in the following commands: DISABLE INTEGRITY CHECKS credit set TEST SIGNING ON Installing the Drivers for the Arduino Uno (from Arduino.cc).
- Plug in your board and wait for Windows to begin it’s driver installation process
- After a few moments, the process will fail, despite its best efforts
- Click on the Start Menu, and open up the Control Panel

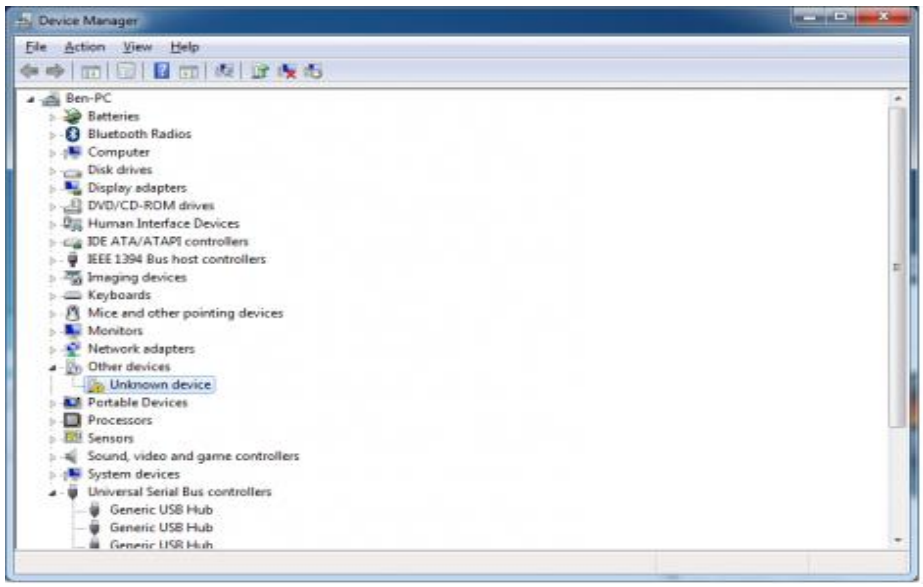


Fig 5.3: Installing Arduino in Windows 8

- Right click on the “Arduino UNO (COMxx)” or “Unknown Device” port and choose the “Update Driver Software” option
- Next, choose the “Browse my computer for Driver software” option
- While in the Control Panel, navigate to System and Security. Next , click on System
- Once the System window is up, open the Device Manager
- Look under Ports (COM & LPT). You should see an open port named “Arduino UNO (COMxx)”. If there is no COM & LPT section, look under ‘Other Devices’ for ‘Unknown Device’

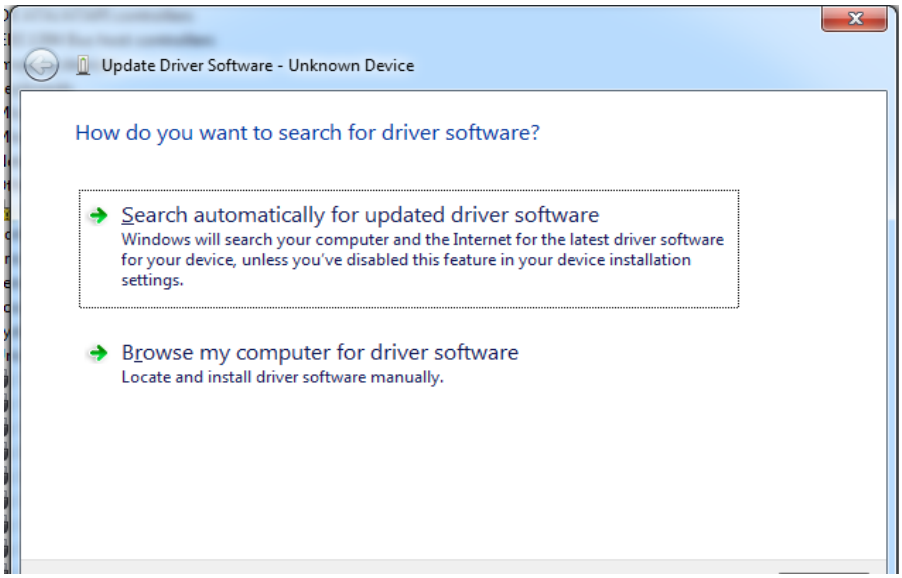


Fig 5.4: Driver Software

- Finally, navigate to and select the Uno’s driver file, named “ArduinoUNO.inf”, located in the “Drivers” folder of the Arduino Software download (not the “FTDI USB Drivers” sub-directory). If you cannot see the in file, it is probably just hidden. You can select the ‘drivers’ folder with the ‘search sub-folders’ option selected instead.
- Windows will finish up the driver installation from there

For earlier versions of the Arduino boards (e.g. Arduino Duemilanove, Nano, or Decimal) check out this page for specific directions. After following the appropriate steps for your software install, we are now ready to test your first program with your Arduino board!

- Launch the Arduino application

- If you disconnected your board, plug it back in
- Open the Blink example sketch by going to: File > Examples > 1.Basics > Blink
- Select the type of Arduino board you're using: Tools > Board > your board type
- The Arduino project provides an integrated development environment (IDE) based on the processing language project.

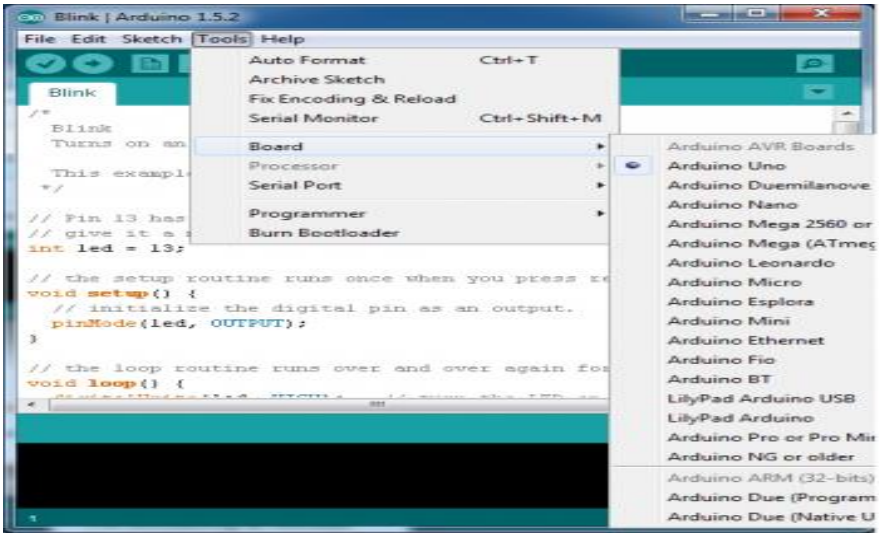


Fig 5.5: Programing in Arduino

- Select the serial/COM port that your Arduino is attached to: Tools > Port > COMxx

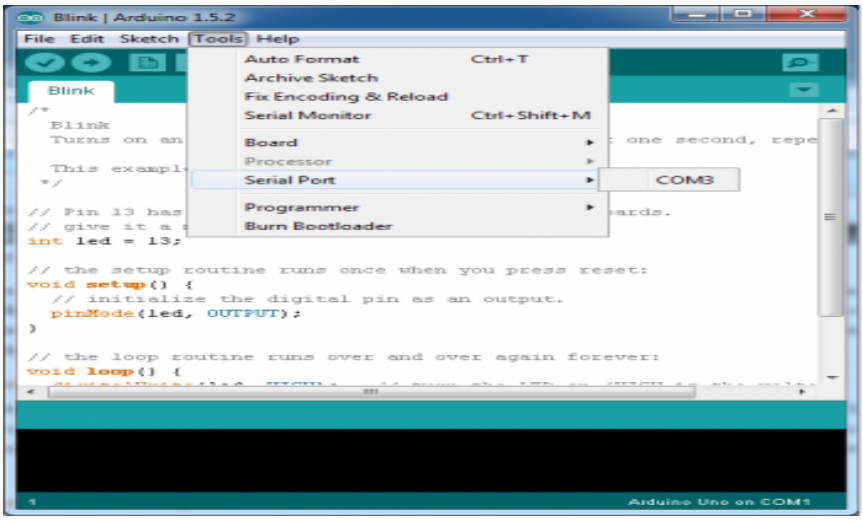


Fig 5.6: Interfacing with ports

- If you're not sure which serial device is your Arduino, take a look at the available ports, then unplug your Arduino and look again. The one that disappeared is your Arduino
- With your Arduino board connected, and the Blink sketch open, press the 'Upload' button

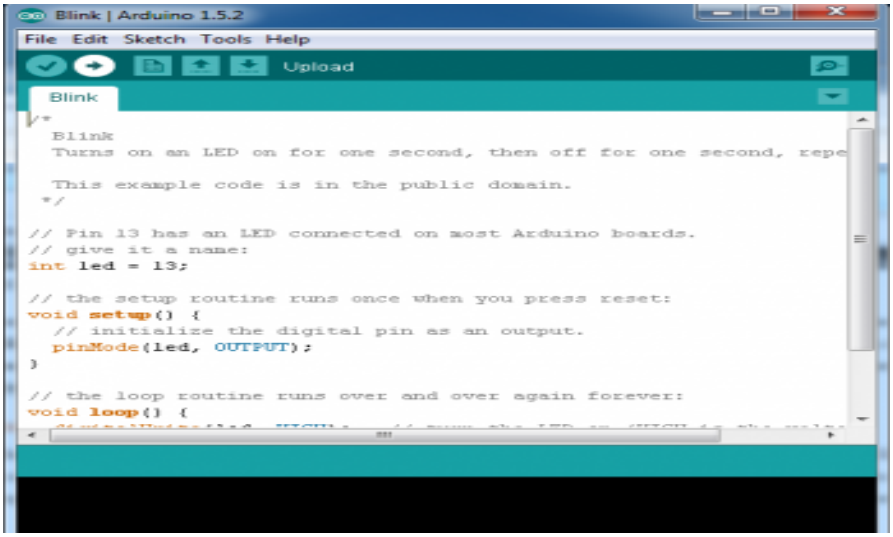


Fig 5.7: Uploading Code into The Board

- After a second, you should see some LEDs flashing on your Arduino, followed by the message 'Done Uploading' in the status bar of the Blink sketch.
- If everything worked, the onboard LED on your Arduino should now be blinking! You just programmed your first Arduino!

5.2 TROUBLESHOOTING

This guide from Arduino has some more details and troubleshooting tips if you get stuck Mac

This page will show you how to install and test the Arduino software on a Mac computer running OSX.

- Go to the Arduino download page and download the latest version of the Arduino software for Mac.
- When the download is finished, un-zip it and open up the Arduino folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is

important so don't be moving any files around unless you really know what you're doing.

- Power up your Arduino by connecting your Arduino board to your computer with a USB cable (or FTDI connector if you're using an Arduino pro). You should see the LED labed 'ON' light up. (This diagram shows the placement of the power LED on the UNO).
- Move the Arduino application into your Applications folder.

5.2.1 FTDI Drivers:

If you have an UNO, Mega2560, or Redboard, you shouldn't need this step, so skip it!

- For other boards, you will need to install drivers for the FTDI chip on your Arduino.
- Go to the FTDI website and download the latest version of the drivers.
- Once you're done downloading, double click the package and follow the instructions from the installer.
- Restart your computer after installing the drivers.

After following the appropriate steps for your software install, we are now ready to test your first program with your Arduino board!

- Launch the Arduino application
- If you disconnected your board, plug it back in
- Open the Blink example sketch by going to: File > Examples > 1.Basics > Blink

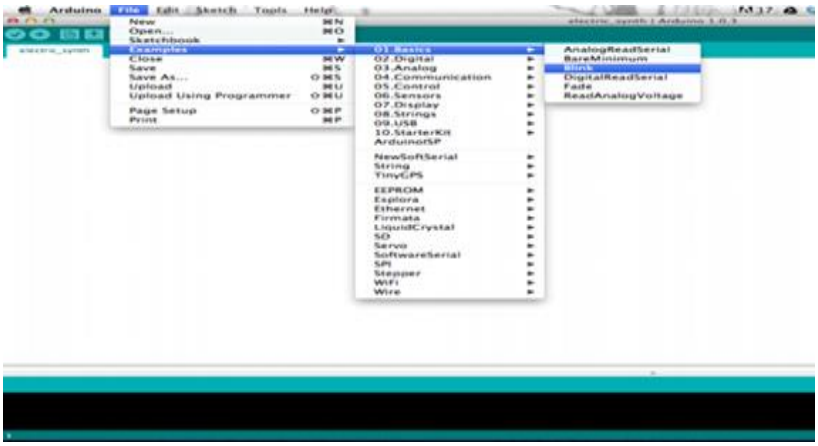


Fig 5.8: FTDI Drivers

- Select the type of Arduino board you're using: Tools > Board > your board type

- Select the serial port that your Arduino is attached to: Tools > Port > xxxxxx (it'll probably look something like "/dev/tty.usbmodemfd131" or "/dev/tty.usbserial-131" but probable with a different number).

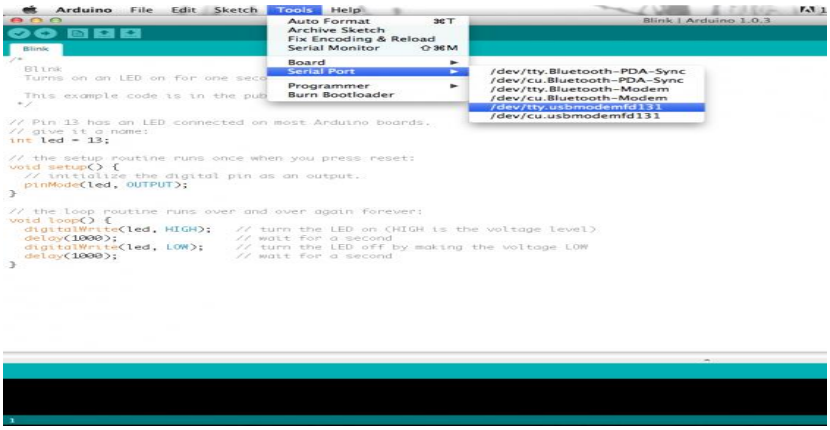


Fig 5.9: Serial Port Attaching

- If you're not sure which serial device is your Arduino, take a look at the available ports, then unplug your Arduino and look again. The one that disappeared is your Arduino.
- With your Arduino board connected and the Blink sketch open, press the 'Upload' button

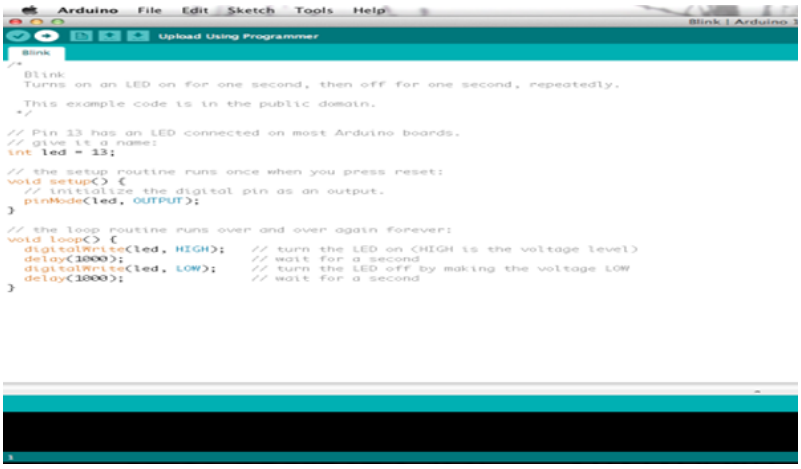


Fig 5.10: Uploading to the Board

- After a second, you should see some LEDs flashing on your Arduino, followed by the message 'Done Uploading' in the status bar of the Blink sketch.
- If everything worked, the onboard LED on your Arduino should now be blinking! You just programmed your first Arduino!

An Arduino is a popular open-source single-board microcontroller. Learn how to program one and let the possibilities take shape.

Sketch

A sketch is a program written with the Arduino IDE. Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.

A minimal Arduino C/C++ program consists of only two functions:

- ***setup*( ):** This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch. It is analogous to the function main( ).
- ***loop*( ):** After setup( ) function exits (ends), the loop( ) function is executed repeatedly in the main program. It controls the board until the board is powered off or is reset. It is analogous to the function while( ).

5.3 Project Software(code) Used in This Project

```
// libraries
#include <MKRGSM.h>
#include "settings.h"

// APN data
String HOLOGRAM_MESSAGE = "";

// initialize the library instance
GSMClient client;
GPRS gprs;
GSM gsmAccess;

// Hologram's Embedded API (https://hologram.io/docs/reference/cloud/embedded/) URL and
port
char server[] = "cloudsocket.hologram.io";
int port = 9999;

const int analogPin=A0; //the AO of the module attach to A0
const int digitalPin=7; //D0 attach to pin7
int Astate=0; //store the value of A0
boolean Dstate=0; //store the value of D0
boolean messageSent = false;
time_t flag;

void setup() {
  pinMode(digitalPin,INPUT); //set digitalPin as INPUT
```

```
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  Astate=analogRead(analogPin); //read the value of A0
  Dstate=digitalRead(digitalPin); //read the value of D0

  {
    if(! messageSent){
      sendNotification();
    }
    else {
      checkHoursLapsed();
    }
  }
  else{
    messageSent = false;
  }
}

void checkHoursLapsed() {
    if (gsmConnect()){
        time_t diff = gsmAccess.getTime() - flag;

        if ((diff / 3600) >= 1) {
          messageSent = false;
          flag = 0;
        }
    }
}

void sendNotification() {

  if (gsmConnect()){
    if (client.connect(server, port)) {
      // Send a Message request:
      HOLOGRAM_MESSAGE = "Heavy Flow detected.";

client.println(HOLOGRAM_PROCEDURE_NAME+HOLOGRAM_DEVICE_KEY+HOLOGRAM_DESTINA
TION_NUMBER+" "+HOLOGRAM_MESSAGE);
      messageSent = true;
      flag = gsmAccess.getTime();
    }

    client.stop();
    gsmAccess.shutdown();
  }
}

boolean gsmConnect() {
  boolean connected = false;
```

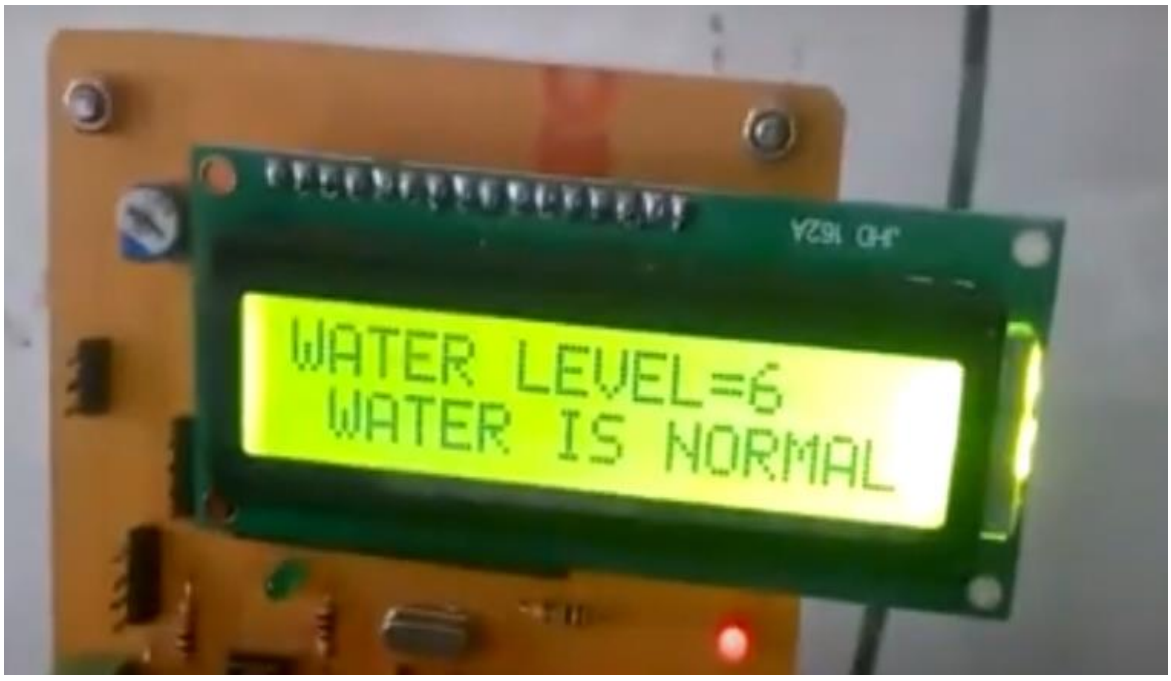
```
  while (!connected) {
    //Serial.println(gsmAccess.begin()); //Uncomment for testing

    if ((gsmAccess.begin() == GSM_READY) &&
        (gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD) == GPRS_READY)) {

          connected = true;
          digitalWrite(LED_BUILTIN, LOW);
    }
    else{
        digitalWrite(LED_BUILTIN, HIGH);
    }
  }

  return connected;
}
```

CHAPTER 6

RESULTS

The prototype of the proposed idea has been implemented using Ultrasonic sensor, GSM, Arduino micro controller and servo motor. The first stage of the implementation was to determine the level of water using ultrasonic sensor. The ultrasonic sensor was mounted on top of a water container to determine the distance between the top of the container and the surface of the water. If the distance goes below a certain point it indicates that the water level in the container has reached a threshold value that is setup and the GSM module sends message to inform the concerned authorities as well as the residents near the dam warning them that the shutters will open soon. After that the shutters are opened by servo motor. When the water level goes below the threshold value that is setup the shutters are closed.

The below figure represents the result of water level.



CHAPTER 7

APPLICATIONS

The above mentioned method will ease the process of water level management on a large scale. We can solve many water related issues by this method. By installing a central command center we are decreasing the manpower required at each and every dam. Since this is a fully automated project, any kind of human intervention has been avoided. So the possibility of faults has also decreased.

In cases of emergency, the override capability will be given to an authorized personnel who can change the command if required. In places where there are issues of water distribution between two areas, this method helps in maintaining neutrality as the command is with the central command center and neither of the areas involved in the fight can give the command.

During times of natural disasters like floods, this method will be very helpful as we don't need to have any human to control near the actual site of the dam. Any command required for the gate opening or gate closing can be given from remote center. This also reduces the response time as the water level data near command center is real time and the decisions are taken almost instantaneously.

Since the data of water levels near all the dams throughout the country are at the same place, a quick decision on the routing of flood water can also be taken. This helps in decreasing the losses due to floods to a significant exten

CHAPTER 8

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- It is easy installation
- Prevent the disaster due to DAM overflow
- Multipurpose
- High Reliability
- Low cost

DISADVANTAGES

- Water level controls need to be replaced every 3years.
- The rust, foul and deteriorate
- No Warranty or Guarantee

CHAPTER 9

CONCLUSION

Water is one of the primary resource for human survival. But unfortunately a mammoth amount of water is being squandered by uncontrolled use. There are certain automated water level monitoring systems in practice but they are used for various applications and have some shortness in practice. We tried to suggest ways to tackle this problem and implement an efficient water level monitoring system using GSM. The main motto of this project work is to establish a flexible, economical and easy configurable system which can solve our water level monitoring problem. among many other issues. We have been using a micro controller to manage the data and to reduce the cost.

FEATURE SCOPE

A cloud based water level monitoring and management network whose flexibility would offer us to control the system from any place via access to cloud data with different type of devices. This type of system is more helpful in situations like floods where the automated gate lifting system will check the water levels and react according the situation. This could have a substantial benefit to the research work related to the efficient management of water at dams by reducing the manual work.

REFERENCES

1. Niteen Mohod, “Usability of Internet of Things [IoT] For Dam Safety and Water Management”, International Journal of Research in Advent Technology, Vol.5, No.1, January 2017.

2. A.C. Khetre, S.G. Hate, “Automatic monitoring & Reporting of water quality by using WSN Technology and different routing methods”, IJARCET Vol 2, Issue 12, Dec 2013, pp 3255- 3260.

3. Amir Ali Khan, Shaden Abdel-Gawad, Haseen Khan, “A real time Water Quality Monitoring Network and Water Quality Indices for River Nile” , abs894_article, IWRA Congress.

4. S.M.Khaled Reza, Shah Ahsanuzzaman, S.M. Mohsin Reza, “Microcontroller Based Automated Water Level Sensing and Controlling: Design and Implementation Issue”, WCECS 2010, October 20-22, 2010, pp 220-224.

5. George Suciu, Adela Vintea, Stefan Ciprian Arseni, Cristina Burca, Victor “Challenges and Solutions for Advanced Sensing of Water Infrastructures in Urban Environments”, 2015 IEEE SIITME , 22-25 Oct 2015, pp 349-352.

6. Samarth Viswanath, Marco Belcastro, John Barton, Brendan O’Flynn, Nicholas Holmes, Paul Dixon, ”Low-power wireless liquid monitoring system using ultrasonic sensors”, IJSSIS, Vol 8, NO.1, March 2015, pp 28-44.

7. Asaad Ahmed Mohammed ahmed Eltaieb1, Zhang Jian Min2 . “Automatic Water Level Control System”, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064.

8. lora-alliance. (2017). lora-alliance. [online] Available at: https://www.lora-alliance.org/technology [Accessed 1 Dec. 2017].

9. En.wikipedia.org. (2017). NarrowBand IOT. [online] Available at: https://en.wikipedia.org/wiki/NarrowBand_IOT [Accessed 1 Dec. 2017]