**CHAPTER 1**

**INTRODUCTION TO EMBEDDED SYSTEMS**

Embedded Technology is now in its prime and the wealth of knowledge available is mind blowing. However, most embedded systems engineers have a common complaint. There are no comprehensive resources available over the internet which deal with the various design and implementation issues of this technology. Intellectual property regulations of many corporations are partly to blame for this and also the tendency to keep technical know-how within a restricted group of researchers.

An embedded computer is frequently a computer that is implemented for a particular purpose. In contrast, an average PC computer usually serves a number of purposes: checking email, surfing the internet, listening to music, word processing, etc... However, embedded systems usually only have a single task, or a very small number of related tasks that they are programmed to perform.

Every home has several examples of embedded computers. Any appliance that has a digital clock, for instance, has a small embedded micro-controller that performs no other task than to display the clock. Modern cars have embedded computers onboard that control such things as ignition timing and anti-lock brakes using input from a number of different sensors.

Embedded computers rarely have a generic interface, however. Even if embedded systems have a keypad and an LCD display, they are rarely capable of using many different types of input or output. An example of an embedded system with I/O capability is a security alarm with an LCD status display, and a keypad for entering a password.

An embedded system can be defined as a control system or computer system designed to perform a specific task. Common examples of embedded systems include MP3 players, navigation systems on aircraft and intruder alarm systems. An embedded system can also be defined as a single purpose computer.

Most embedded systems are time critical applications meaning that the embedded system is working in an environment where timing is very important: the results of an operation are only relevant if they take place in a specific time frame. An autopilot in an aircraft is a time critical embedded system. If the autopilot detects that the plane for some reason is going into a stall then it should take steps to correct this within milliseconds or there would be catastrophic results.

* 1. **APPLICATIONS OF EMBEDDED SYSTEM**

Embedded systems are commonly found in consumer, cooking, industrial, automotive, medical, commercial and military applications.

Telecommunications systems employ numerous embedded systems from [telephone switches](https://en.wikipedia.org/wiki/Telephone_switch) for the network to [cell phones](https://en.wikipedia.org/wiki/Cell_phone) at the end user. Computer networking uses dedicated [routers](https://en.wikipedia.org/wiki/Router_(computing)) and [network bridges](https://en.wikipedia.org/wiki/Network_bridge) to route data.

[Consumer electronics](https://en.wikipedia.org/wiki/Consumer_electronics) include [MP3 players](https://en.wikipedia.org/wiki/MP3_player), mobile phones, [videogame consoles](https://en.wikipedia.org/wiki/Videogame_console), [digital cameras](https://en.wikipedia.org/wiki/Digital_camera), [GPS](https://en.wikipedia.org/wiki/Global_Positioning_System) receivers, and [printers](https://en.wikipedia.org/wiki/Computer_printer). Household appliances, such as [microwave ovens](https://en.wikipedia.org/wiki/Microwave_oven), [washing machines](https://en.wikipedia.org/wiki/Washing_machine) and [dishwashers](https://en.wikipedia.org/wiki/Dishwashers), include embedded systems to provide flexibility, efficiency and features. Advanced [HVAC](https://en.wikipedia.org/wiki/HVAC) systems use networked [thermostats](https://en.wikipedia.org/wiki/Thermostat) to more accurately and efficiently control temperature that can change by time of day and [season](https://en.wikipedia.org/wiki/Season). [Home automation](https://en.wikipedia.org/wiki/Home_automation) uses wired- and wireless-networking that can be used to control lights, climate, security, audio/visual, surveillance, etc., all of which use embedded devices for sensing and controlling.

Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced [avionics](https://en.wikipedia.org/wiki/Avionics) such as [inertial guidance systems](https://en.wikipedia.org/wiki/Inertial_guidance_system) and [GPS](https://en.wikipedia.org/wiki/Global_Positioning_System) receivers that also have considerable safety requirements. Various electric motors [brushless DC motors](https://en.wikipedia.org/wiki/Brushless_DC_motor), [induction motors](https://en.wikipedia.org/wiki/Induction_motor) and [DC motors](https://en.wikipedia.org/wiki/DC_motor)  use electric/electronic [motor controllers](https://en.wikipedia.org/wiki/Motor_controller). [Automobiles](https://en.wikipedia.org/wiki/Automobile), [electric vehicles](https://en.wikipedia.org/wiki/Electric_vehicle), and [hybrid vehicles](https://en.wikipedia.org/wiki/Hybrid_vehicle) increasingly use embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems include [anti-lock braking system](https://en.wikipedia.org/wiki/Anti-lock_braking_system) (ABS), [Electronic Stability Control](https://en.wikipedia.org/wiki/Electronic_Stability_Control) (ESC/ESP), [traction control](https://en.wikipedia.org/wiki/Traction_control_system) (TCS) and automatic [four-wheel drive](https://en.wikipedia.org/wiki/Four-wheel_drive).

[Medical equipment](https://en.wikipedia.org/wiki/Medical_equipment) uses embedded systems for [vital signs](https://en.wikipedia.org/wiki/Vital_signs) monitoring, [electronic stethoscopes](https://en.wikipedia.org/wiki/Electronic_stethoscope) for amplifying sounds, and various [medical imaging](https://en.wikipedia.org/wiki/Medical_imaging) ([PET](https://en.wikipedia.org/wiki/Positron_emission_tomography), [SPECT](https://en.wikipedia.org/wiki/Single_photon_emission_computed_tomography), [CT](https://en.wikipedia.org/wiki/Computed_tomography), and [MRI](https://en.wikipedia.org/wiki/Magnetic_resonance_imaging)) for non-invasive internal inspections. Embedded systems within medical equipment are often powered by industrial computers.[[9]](https://en.wikipedia.org/wiki/Embedded_system#cite_note-9)

Embedded systems are used in transportation, fire safety, safety and security, medical applications and life critical systems, as these systems can be isolated from hacking and thus, be more reliable. For fire safety, the systems can be designed to have greater ability to handle higher temperatures and continue to operate. In dealing with security, the embedded systems can be self-sufficient and be able to deal with cut electrical and communication systems.

* 1. **CHARACTERISTICS OF EMBEDDED SYSTEM**

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have [real-time](https://en.wikipedia.org/wiki/Real-time_computing) performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

Embedded systems are not always standalone devices. Many embedded systems consist of small parts within a larger device that serves a more general purpose. For example, the [Gibson Robot Guitar](https://en.wikipedia.org/wiki/Gibson_Robot_Guitar) features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music. Similarly, an embedded system in an [automobile](https://en.wikipedia.org/wiki/Automobile) provides a specific function as a subsystem of the car itself.

Embedded systems range from [no user interface](https://en.wikipedia.org/wiki/Headless_system) at all, in systems dedicated only to one task, to complex [graphical user interfaces](https://en.wikipedia.org/wiki/Desktop_operating_system#Graphical_user_interfaces) that resemble modern computer desktop operating systems. Simple embedded devices use [buttons](https://en.wikipedia.org/wiki/Push-button), [LEDs](https://en.wikipedia.org/wiki/LED), graphic or character [LCDs](https://en.wikipedia.org/wiki/LCD) ([HD44780 LCD](https://en.wikipedia.org/wiki/Hitachi_HD44780_LCD_controller) for example) with a simple [menu system](https://en.wikipedia.org/wiki/Menu_(computing)).

More sophisticated devices which use a graphical screen with [touch](https://en.wikipedia.org/wiki/Touch_screen) sensing or screen-edge buttons provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what is desired. [Handheld systems](https://en.wikipedia.org/wiki/Mobile_device) often have a screen with a "joystick button" for a pointing device.

Some systems provide user interface remotely with the help of a serial (e.g. [RS-232](https://en.wikipedia.org/wiki/RS-232), [USB](https://en.wikipedia.org/wiki/USB), [I²C](https://en.wikipedia.org/wiki/I%C2%B2C), etc.) or network (e.g. [Ethernet](https://en.wikipedia.org/wiki/Ethernet)) connection. This approach gives several advantages: extends the capabilities of embedded system, avoids the cost of a display, simplifies [BSP](https://en.wikipedia.org/wiki/Board_support_package) and allows one to build a rich user interface on the PC. A good example of this is the combination of an [embedded web server](https://en.wikipedia.org/wiki/Embedded_HTTP_server) running on an embedded device (such as an [IP camera](https://en.wikipedia.org/wiki/IP_camera)) or a [network router](https://en.wikipedia.org/wiki/Router_(computing)). The user interface is displayed in a [web browser](https://en.wikipedia.org/wiki/Web_browser) on a PC connected to the device, therefore needing no software to be installed.

* 1. **PROCESSORS IN EMBEDDED SYSTEMS**

Embedded processors can be broken into two broad categories. Ordinary microprocessors Embedded processors can be broken into two broad categories. Ordinary microprocessors (μP) use separate integrated circuits for memory and peripherals. Microcontrollers (μC) have on-chip peripherals, thus reducing power consumption, size and cost. In contrast to the personal computer market, many different basic [CPU architectures](https://en.wikipedia.org/wiki/CPU_architecture) are used, since software is custom-developed for an application and is not a commodity product installed by the end user. Both [Von Neumann](https://en.wikipedia.org/wiki/Von_Neumann_architecture) as well as various degrees of [Harvard architectures](https://en.wikipedia.org/wiki/Harvard_architecture) are used. [RISC](https://en.wikipedia.org/wiki/RISC) as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits and beyond, although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

[Numerous microcontrollers](https://en.wikipedia.org/wiki/List_of_common_microcontrollers) have been developed for embedded systems use. General-purpose microprocessors are also used in embedded systems, but generally require more support circuitry than microcontrollers.

(μP) use separate integrated circuits for memory and peripherals. Microcontrollers (μC) have on-chip peripherals, thus reducing power consumption, size and cost. In contrast to the personal computer market, many different basic [CPU architectures](https://en.wikipedia.org/wiki/CPU_architecture) are used, since software is custom-developed for an application and is not a commodity product installed by the end user. Both [Von Neumann](https://en.wikipedia.org/wiki/Von_Neumann_architecture) as well as various degrees of [Harvard architectures](https://en.wikipedia.org/wiki/Harvard_architecture) are used. [RISC](https://en.wikipedia.org/wiki/RISC) as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits and beyond, although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

[Numerous microcontrollers](https://en.wikipedia.org/wiki/List_of_common_microcontrollers) have been developed for embedded systems use. General-purpose microprocessors are also used in embedded systems, but generally require more support circuitry than microcontrollers.

**1.4 DEBUGGING IN EMBEDDED SYSTEMS**

Embedded [debugging](https://en.wikipedia.org/wiki/Debugging) may be performed at different levels, depending on the facilities available. The different metrics that characterize the different forms of embedded debugging are: does it slow down the main application, how close is the debugged system or application to the actual system or application, how expressive are the triggers that I can set for debugging (e.g., I want to inspect the memory when a particular [program counter](https://en.wikipedia.org/wiki/Program_counter) value is reached), and what can I inspect in the debugging process (such as, only memory, or memory and registers, etc.).

From simplest to most sophisticated 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](https://en.wikipedia.org/wiki/Remedy_Debugger) which even works for heterogeneous [multicore](https://en.wikipedia.org/wiki/Multi-core_processor) systems.

An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a [JTAG](https://en.wikipedia.org/wiki/JTAG) or [Nexus](https://en.wikipedia.org/wiki/Nexus_(standard)) 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](https://en.wikipedia.org/wiki/In-circuit_emulator) (ICE) replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.

A complete [emulator](https://en.wikipedia.org/wiki/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. The downsides are expense and slow operation, in some cases up to 100 times slower than the final system.

For SoC designs, the typical approach is to verify and debug the design on an FPGA prototype board. Tools such as Certus[[11]](https://en.wikipedia.org/wiki/Embedded_system#cite_note-12) are used to insert probes in the FPGA RTL that make signals available for observation. This is used to debug hardware, firmware and software interactions across multiple FPGA with capabilities similar to a logic analyzer.

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 [HLL](https://en.wikipedia.org/wiki/High-level_programming_language) [source-code](https://en.wikipedia.org/wiki/Source-code), [assembly code](https://en.wikipedia.org/wiki/Assembly_code) or mixture of both.

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, and 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](https://en.wikipedia.org/wiki/Logic_analyzer), for instance.

**1.5 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 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.
* 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](https://en.wikipedia.org/wiki/Memory_leak), and also [soft errors](https://en.wikipedia.org/wiki/Soft_error) in the hardware:

* [watchdog timer](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Trusted_Computing_Base) (TCB) architecture[[12]](https://en.wikipedia.org/wiki/Embedded_system#cite_note-13) ensures a highly secure & reliable system environment
* A [hypervisor](https://en.wikipedia.org/wiki/Hypervisor) designed for embedded systems, 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](https://en.wikipedia.org/wiki/Immunity_Aware_Programming)

**1.6 TRACING**

Real-time operating systems ([RTOS](https://en.wikipedia.org/wiki/RTOS)) often supports [tracing](https://en.wikipedia.org/wiki/Tracing_(software)) of operating system events. A graphical view is presented by a host PC tool, based on a recording of the system behavior. The trace recording can be performed in software, by the RTOS, or by special tracing hardware. RTOS tracing allows developers to understand timing and performance issues of the software system and gives a good understanding of the high-level system behaviors. Commercial tools like [RTXC Quadros](https://en.wikipedia.org/wiki/RTXC_Quadros) or [IAR Systems](https://en.wikipedia.org/wiki/IAR_Systems) exists.

**CHAPTER 2**

**FOOT STEP POWER GENERATION WITH RFID MOBILE CHARGING**

**2.1 INTRODUCTION**

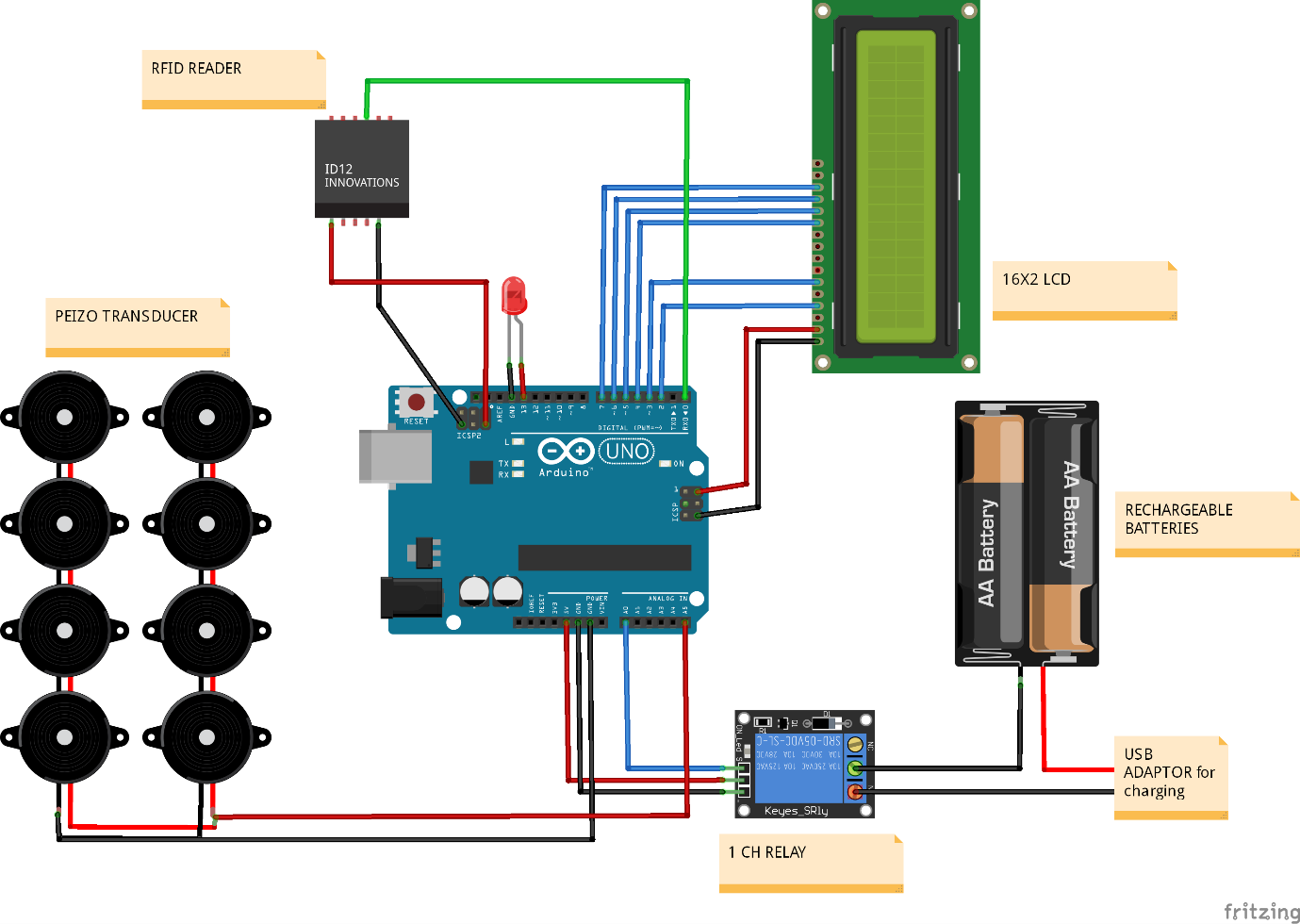
Here we propose an advanced footstep power generator system that uses peizo sensors to generate power from human footsteps.

The system allows for a platform for placing footsteps. The peizo sensors are mounted below the platform to generate voltage from footsteps.The sensors are placed in such an arrangement so as to generate maximum output voltage. This is then provided to our monitoring circuitry. The circuit is a microcontroller based monitoring circuit that allows user to monitor the voltage and charges a connected battery by it.

It also displays the charge generated and displays on an LCD display. Also it consists of a USB mobile phone charging point where user may connect cables to charge mobile phone from the battery charge. Thus we charge a battery using power from user footsteps, display it on lcd using microcontroller circuit and allow for mobile charging through the setup.

Day by day, the population of the country is increasing and the requirement of the power is also increasing. At the same time the wastage of energy is also increasing in many ways. So, reforming this energy back to a usable form is the major solution. In this footstep power generation project, we are generating power with the help of human’s footsteps; this power is then used to charge battery. The power is stored in a battery that can be used to charge a mobile phone using RFID card. This system is powered by Atmega 328 microcontroller, it consists of Arduino IDE, RFID sensor, USB cable and LCD. When we power on the system, the system enters into registration mode. We can register three users. Once all the user is entered in the system then the system asks to swipe the card and connect the charger. Initially all the user is given 5 minutes of charging time as default. When we swipe the card and if the user is authorised, the system turns on for charging and will charge the Mobile phone. If the user is un-authorised then the system will display as unauthorised user, just in case if the user wants to stop the charging in midway the user needs to swipe the card again. As soon as the card is swiped again, the remaining time balance is displayed and the charging stops. In order to recharge a card, we need to press recharge button which is on the system, and then system will ask to swipe the card, once the user swipes the card, it adds more 5 minutes to the particular card of the user.

**2.2 CIRCUIT DIAGRAM**

****

**CHAPTER 3**

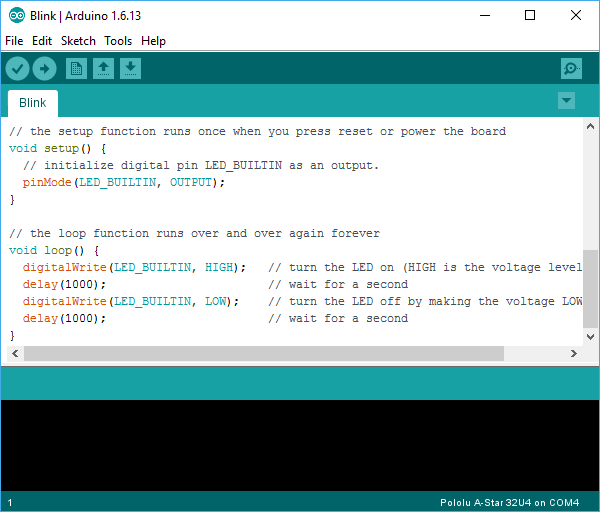
**SOFTWARE REQUIREMENTS**

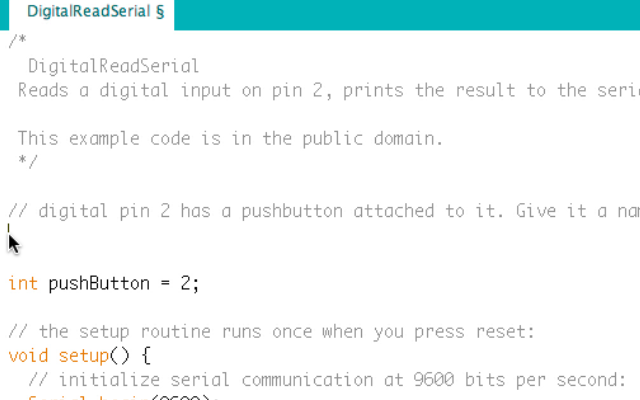
Software used in this project for uploading code onto Arduino is Arduino IDE.

**3.1 INTRODUCTION TO ARDUINO IDE**

IDE stands for Integrated Development Environment. Pretty fancy sounding, and should make you feel smart any time you use it. The IDE is a text editor-like program that allows you to write Arduino code. When you open the Arduino program, you are opening the IDE. It is intentionally streamlined to keep things as simple and straightforward as possible. When you save a file in Arduino, the file is called a sketch – a sketch is where you save the computer code you have written. The coding language that Arduino uses is very much like C++ (“see plus plus”), which is a common language in the world of computing. The code you learn to write for Arduino will be very similar to the code you write in any other computer language – all the basic concepts remain the same – it is just a matter of learning a new dialect should you pursue other programming languages.



The code you write is “human readable”, that is, it will make sense to you (sometimes), and will be organized for a human to follow. Part of the job of the IDE is to take the human readable code and translate it into machine-readable code to be executed by the Arduino. This process is called compiling. The process of compiling is seamless to the user. All you have to do is press a button. If you have errors in your computer code, the compiler will display an error message at the bottom of the IDE and highlight the line of code that seems to be the issue. The error message is meant to help you identify what you might have done wrong – sometimes the message is very explicit, like saying, “Hey – you forget a semicolon”, sometimes the error message is vague. Why be concerned with a semicolon you ask? A semicolon is part of the Arduino language syntax, the rules that govern how the code is written. It is like grammar in writing. Say for example we didn’t use periods when we wrote – everyone would have a heck of a time trying to figure out when sentences started and ended. Or if we didn’t employ the comma, how would we convey a dramatic pause to the reader?

And let me tell you, if you ever had an English teacher with an overactive red pen, the compiler is ten times worse. In fact – your programs WILL NOT compile without perfect syntax. This might drive you crazy at first because it is very natural to forget syntax. As you gain experience programming you will learn to be assiduous about coding grammar.

**3.1.1 THE SEMICOLON**

A semicolon needs to follow every statement written in the Arduino programming language. For example, …

Int LedPin=9;

In this statement, I am assigning a value to an integer variable (we will cover this later), notice the semicolon at the end. This tells the compiler that you have finished a chunk of code and are moving on to the next piece. A semicolon is to Arduino code, as a period is to a sentence. It signifies a complete statement.

### 3.1.2 THE DOUBLE BACKSLASH FOR SINGLE LINE COMMENTS //

### Comments are what you use to annotate code. Good code is commented well. Comments are meant to inform you and anyone else who might stumble across your code, what the heck you were thinking when you wrote it. A good comment would be something like this…

Now, in 3 months when I review this program, I know where to stick my LED. Comments will be ignored by the compiler – so you can write whatever you like in them. If you have a lot you need to explain, you can use a multi-line comment, shown below…

//This is an example

Comments are like the footnotes of code, except far more prevalent and not at the bottom of the page.

**3.1.3 THE CURLY BRACES**

Curly braces are used to enclose further instructions carried out by a function (we discuss functions next). There is always an opening curly bracket and a closing curly bracket. If you forget to close a curly bracket, the compiler will not like it and throw an error code.

Void loop(){

}

Remember – no curly brace may go unclosed!

**3.1.4 FUNCTION ( )**

Functions are pieces of code that are used so often that they are encapsulated in certain keywords so that you can use them more easily. For example, a function could be the following set of instructions…

This set of simple instructions could be encapsulated in a function that we call WashDog. Every time we want to carry out all those instructions we just type WashDog and voila – all the instructions are carried out. In Arduino, there are certain functions that are used so often they have been built into the IDE. When you type them, the name of the function will appear orange. The function pinMode(), for example, is a common function used to designate the mode of an Arduino pin.

What’s the deal with the parentheses following the function pinMode? Many functions require *arguments* to work. An argument is information the function uses when it runs. For our WashDog function, the arguments might be dog name and soap type, or temperature and size of a bucket.

pinMode(13,OUTPUT);

The argument 13 refers to pin 13, and OUTPUT is the mode in which you want the pin to operate. When you enter these arguments the terminology is called passing. You pass the necessary information to the functions. Not all functions require arguments, but opening and closing parentheses will stay regardless though empty.

Notice that the word OUTPUT is blue. There are certain keywords in Arduino that are used frequently and the color blue helps identify them. The IDE turns them blue automatically. Now we won’t get into it here, but you can easily make your own functions in Arduino, and you can even get the IDE to color them for you. We will, however, talk about the two functions used in nearly EVERY Arduino program.

**3.1.5 VOID SETUP ( )**

The function, setup(), as the name implies, is used to set up the Arduino board. The Arduino executes all the code that is contained between the curly braces of setup() only once. Typical things that happen in setup() are setting the modes of pins, starting You might be wondering what void means before the function setup(). Void means that the function does not return information. Some functions do return values – our DogWash function might return the number of buckets it required to clean the dog. The function analogRead() returns an integer value between 0-1023. If this seems a bit odd now, don’t worry as we will cover every common Arduino function in depth as we continue the course.

Let us review a couple things you should know about setup()…

1. setup() only runs once.

2. setup() needs to be the first function in your Arduino sketch.

3. setup() must have opening and closing curly braces.

**3.1.6 VOID LOOP ( )**

You have to love the Arduino developers because the function names are so telling. As the name implies, all the code between the curly braces in loop() is repeated over and over again – in a loop. The loop() function is where the body of your program will reside. As with setup(), the function loop() does not return any values, therefore the word void precedes it.

Does it seem odd to you that the code runs in one big loop? This apparent lack of variation is an illusion. Most of your code will have specific conditions laying in wait which will trigger new actions.

If you have a temperature sensor connected to your Arduino for example, then when the temperature gets to a predefined threshold you might have a fan kick on. The looping code is constantly checking the temperature waiting to trigger the fan. So even though the code loops over and over, not every piece of the code will be executed every iteration of the loop.

**3.2 INTRODUCTION ARDUINO LIBRARIES**

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are [listed in the reference](https://www.arduino.cc/en/Reference/Libraries). To use the additional libraries, you will need to install them.

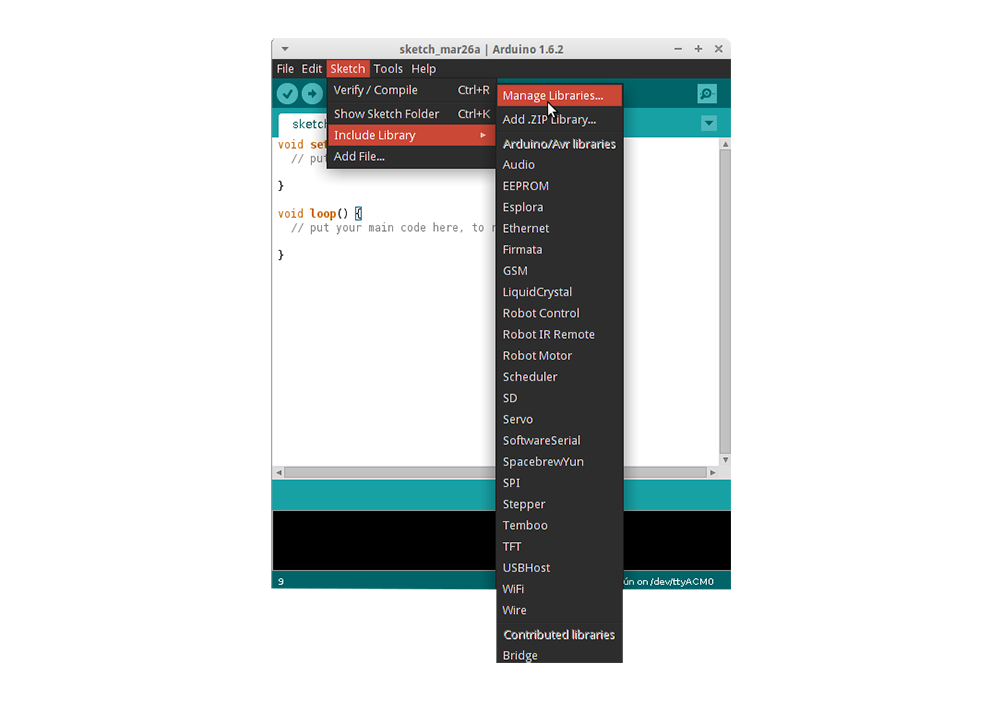
Arduino libraries are managed in three different places: inside the IDE installation folder, inside the core folder and in the libraries folder inside your sketchbook. The way libraries are chosen during compilation is designed to allow the update of libraries present in the distribution. This means that placing a library in the “libraries” folder in your sketchbook overrides the other libraries versions.

The same happens for the libraries present in additional cores installations. It is also important to note that the version of the library you put in your sketchbook may be lower than the one in the distribution or core folders, nevertheless it will be the one used during compilation. When you select a specific core for your board, the libraries present in the core’s folder are used instead of the same libraries present in the IDE distribution folder.

Last, but not least important is the way the Arduino Software (IDE) upgrades itself: all the files in Programs/Arduino (or the folder where you installed the IDE) are deleted and a new folder is created with fresh content. This is why we recommend that you only install libraries to the sketchbook folder so they are not deleted during the Arduino IDE update process.

**3.2.1 HOW TO INSTALL A LIBRARY**

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



Then the Library Manager will open and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, click on it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, don't worry: it is normal.



Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager. You can now find the new library available in the Sketch > Include Library menu. If you want to add your own library to Library Manager, follow [these instructions](https://github.com/arduino/Arduino/wiki/Library-Manager-FAQ).

**3.3 HOW TO CONNECT ARDUINO BOARD**

If you're using a serial board, power the board with an external power supply (6 to 25 volts DC, with the core of the connector positive). Connect the board to a serial port on your computer. On the USB boards, the power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (needed for motors and other high current devices), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer. On Windows, the Add New Hardware wizard will open; tell it you want to specify the location to search for drivers and point to the folder containing the USB drivers you unzipped in the previous step.

The power LED should go on.

**3.4 HOW TO UPLOAD A PROGRAM**

The content of circuits and Arduino sketches can vary greatly. Before you get started, there is one simple process for uploading a sketch to an Arduino board that you can refer back to.

Follow these steps to upload your sketch:

1. Connect your Arduino using the USB cable.

The square end of the USB cable connects to your Arduino and the flat end connects to a USB port on your computer.

1. Choose Tools→Board→Arduino Uno to find your board in the Arduino menu.

You can also find all boards through this menu, such as the Arduino MEGA 2560 and Arduino Leonardo.

1. Choose the correct serial port for your board.

You find a list of all the available serial ports by choosing Tools→Serial Port→ comX or /dev/tty.usbmodemXXXXX. X marks a sequentially or randomly assigned number. In Windows, if you have just connected your Arduino, the COM port will normally be the highest number, such as com 3 or com 15.

Many devices can be listed on the COM port list, and if you plug in multiple Arduinos, each one will be assigned a new number. On Mac OS X, the /dev/tty.usbmodem number will be randomly assigned and can vary in length, such as /dev/tty.usbmodem1421 or /dev/tty.usbmodem262471. Unless you have another Arduino connected, it should be the only one visible.

1. Click the Upload button.

This is the button that points to the right in the Arduino environment. You can also use the keyboard shortcut Ctrl+U for Windows or Cmd+U for Mac OS X.

**CHAPTER 4**

**HARDWARE REQUIREMENTS**

Hardware Components of this project are

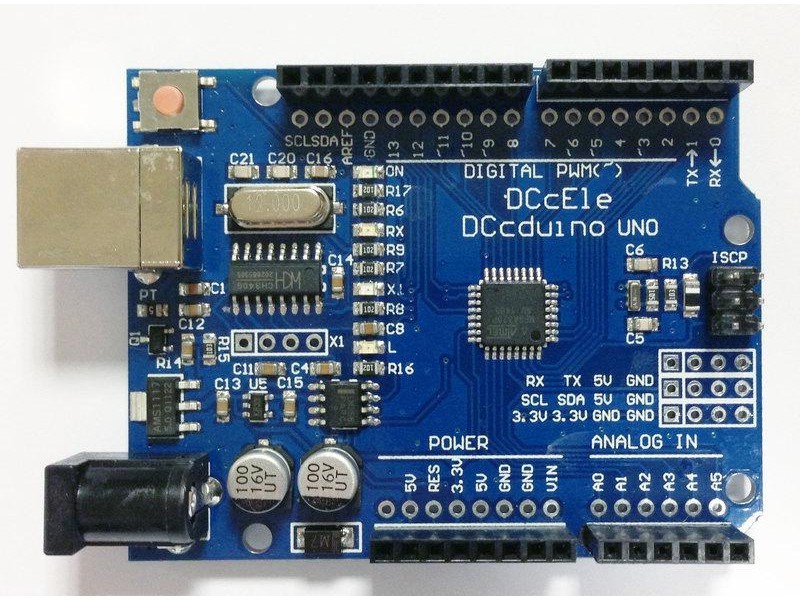
* 1. Arduino uno
  2. 16X 2 LCD
  3. RELAY
  4. PEIZO ELECTRIC TRANSDUCER
  5. Rfid reader
  6. 18650 LI ION BATTERY

**4.1 INTRODUCTION TO ARDUINO UNO**

Arduino is an open-source electronics platform based on easy-to-use hardware and software. [Arduino boards](https://www.arduino.cc/en/Main/Products) 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](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

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](http://forum.arduino.cc/) 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. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The [software](https://www.arduino.cc/en/Main/Software), too, is open-source, and it is growing through the contributions of users worldwide.



**4.1.1 WHY ARDUINO**

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.

**4.1.2 ADVANTAGES OF ARDUINO**

* **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
* **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
* **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
* **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
* **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the [breadboard version of the module](https://www.arduino.cc/en/Main/Standalone) in order to understand how it works and save money.

**4.1.3 FEATURES OF ARDUINO UNO**

The **Arduino Uno** is a microcontroller board based on the ATmega328. Arduino is an open-source, prototyping platform and its simplicity makes it ideal for  hobbyists to use as well as professionals. The Arduino Uno 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.

**Features of the Arduino UNO:**

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

**4.2 INTRODUCTION TO 16X2 LCD MODULE**

A liquid-crystal display (LCD) is a [flat-panel display](https://en.wikipedia.org/wiki/Flat_panel_display) or other [electronically modulated optical device](https://en.wikipedia.org/wiki/Electro-optic_modulator) that uses the light-modulating properties of [liquid crystals](https://en.wikipedia.org/wiki/Liquid_crystal). Liquid crystals do not emit light directly, instead using a [backlight](https://en.wikipedia.org/wiki/Backlight) or [reflector](https://en.wikipedia.org/wiki/Reflector_(photography)) to produce images in color or [monochrome](https://en.wikipedia.org/wiki/Monochrome).[[1]](https://en.wikipedia.org/wiki/Liquid-crystal_display#cite_note-1) LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and [7-segment](https://en.wikipedia.org/wiki/7-segment) displays, as in a [digital clock](https://en.wikipedia.org/wiki/Digital_clock). They use the same basic technology, except that arbitrary images are made up of a large number of small [pixels](https://en.wikipedia.org/wiki/Pixel), while other displays have larger elements.

LCDs are used in a wide range of applications including [computer monitors](https://en.wikipedia.org/wiki/Computer_monitor), [televisions](https://en.wikipedia.org/wiki/Television), [instrument panels](https://en.wikipedia.org/wiki/Dashboard), [aircraft cockpit displays](https://en.wikipedia.org/wiki/Flight_instruments), and indoor and outdoor signage. Small LCD screens are common in portable consumer devices such as [digital cameras](https://en.wikipedia.org/wiki/Digital_camera), [watches](https://en.wikipedia.org/wiki/Watch), [calculators](https://en.wikipedia.org/wiki/Calculator), and [mobile telephones](https://en.wikipedia.org/wiki/Mobile_telephone), including [smartphones](https://en.wikipedia.org/wiki/Smartphone). LCD screens are also used on [consumer electronics](https://en.wikipedia.org/wiki/Consumer_electronics) products such as DVD players, video game devices and [clocks](https://en.wikipedia.org/wiki/Clock). LCD screens have replaced heavy, bulky [cathode ray tube](https://en.wikipedia.org/wiki/Cathode_ray_tube) (CRT) displays in nearly all applications. LCD screens are available in a wider range of screen sizes than CRT and [plasma displays](https://en.wikipedia.org/wiki/Plasma_display), with LCD screens available in sizes ranging from tiny [digital watches](https://en.wikipedia.org/wiki/Digital_watch) to huge, big-screen [television sets](https://en.wikipedia.org/wiki/Television_set).

Since LCD screens do not use phosphors, they do not suffer [image burn-in](https://en.wikipedia.org/wiki/Screen_burn-in) when a static image is displayed on a screen for a long time (e.g., the table frame for an aircraft schedule on an indoor sign). LCDs are, however, susceptible to [image persistence](https://en.wikipedia.org/wiki/Image_persistence).[[2]](https://en.wikipedia.org/wiki/Liquid-crystal_display#cite_note-Fujitsu-2) The LCD screen is more energy-efficient and can be disposed of more safely than a CRT can. Its low electrical power consumption enables it to be used in [battery](https://en.wikipedia.org/wiki/Battery_(electricity))-powered [electronic](https://en.wikipedia.org/wiki/Electronics) equipment more efficiently than CRTs can be. By 2008, annual sales of televisions with LCD screens exceeded sales of CRT units worldwide, and the CRT became obsolete for most purposes.



A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a [LCD](http://www.engineersgarage.com/insight/how-lcd-works).



**4.2.1 LCD PIN DESCRIPTION**

|  |  |  |
| --- | --- | --- |
| **Pin No** | **Function** | **Name** |
| 1 | Ground (0V) | Ground |
| 2 | Supply voltage; 5V (4.7V – 5.3V) | Vcc |
| 3 | Contrast adjustment; through a variable resistor | VEE |
| 4 | Selects command register when low; and data register when high | Register Select |
| 5 | Low to write to the register; High to read from the register | Read/write |
| 6 | Sends data to data pins when a high to low pulse is given | Enable |
| 7 | 8-bit data pins | DB0 |
| 8 | DB1 |
| 9 | DB2 |
| 10 | DB3 |
| 11 | DB4 |
| 12 | DB5 |
| 13 | DB6 |
| 14 | DB7 |
| 15 | Backlight VCC (5V) | Led+ |
| 16 | Backlight Ground (0V) | Led- |

**4.3 INTRODUCTION TO RELAY**

A **relay** is an [electrically](https://en.wikipedia.org/wiki/Electric) operated [switch](https://en.wikipedia.org/wiki/Switch). Many relays use an [electromagnet](https://en.wikipedia.org/wiki/Electromagnet) to mechanically operate a switch, but other operating principles are also used, such as [solid-state relays](https://en.wikipedia.org/wiki/Solid-state_relay). Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long distance [telegraph](https://en.wikipedia.org/wiki/Electrical_telegraph) circuits as amplifiers: they repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a [contactor](https://en.wikipedia.org/wiki/Contactor). [Solid-state relays](https://en.wikipedia.org/wiki/Solid-state_relay) control power circuits with no [moving parts](https://en.wikipedia.org/wiki/Moving_parts), instead using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digital instruments still called "[protective relays](https://en.wikipedia.org/wiki/Protective_relay)".

Magnetic latching relays require one pulse of coil power to move their contacts in one direction, and another, redirected pulse to move them back. Repeated pulses from the same input have no effect. Magnetic latching relays are useful in applications where interrupted power should not be able to transition the contacts.

Magnetic latching relays can have either single or dual coils. On a single coil device, the relay will operate in one direction when power is applied with one polarity, and will reset when the polarity is reversed. On a dual coil device, when polarized voltage is applied to the reset coil the contacts will transition. AC controlled magnetic latch relays have single coils that employ steering diodes to differentiate between operate and reset commands.



**4.3.1 BASIC DESIGN AND OPERATION**

A simple electromagnetic relay consists of a coil of wire wrapped around a [soft iron core](https://en.wikipedia.org/wiki/Magnetic_core), an iron yoke which provides a low [reluctance](https://en.wikipedia.org/wiki/Magnetic_reluctance) path for magnetic flux, a movable iron [armature](https://en.wikipedia.org/wiki/Armature_(electrical_engineering)), and one or more sets of contacts (there are two contacts in the relay pictured). The armature is hinged to the yoke and mechanically linked to one or more sets of moving contacts. The armature is held in place by a [spring](https://en.wikipedia.org/wiki/Spring_(device)) so that when the relay is de-energized there is an air gap in the magnetic circuit. In this condition, one of the two sets of contacts in the relay pictured is closed, and the other set is open. Other relays may have more or fewer sets of contacts depending on their function. The relay in the picture also has a wire connecting the armature to the yoke. This ensures continuity of the circuit between the moving contacts on the armature, and the circuit track on the [printed circuit board](https://en.wikipedia.org/wiki/Printed_circuit_board) (PCB) via the [yoke](https://en.wikipedia.org/wiki/Yoke), which is soldered to the PCB.

When an [electric current](https://en.wikipedia.org/wiki/Electric_current) is passed through the coil it generates a [magnetic field](https://en.wikipedia.org/wiki/Magnetic_field) that activates the armature, and the consequent movement of the movable contact(s) either makes or breaks (depending upon construction) a connection with a fixed contact. If the set of contacts was closed when the relay was de-energized, then the movement opens the contacts and breaks the connection, and vice versa if the contacts were open. When the current to the coil is switched off, the armature is returned by a force, approximately half as strong as the magnetic force, to its relaxed position. Usually this force is provided by a spring, but gravity is also used commonly in industrial motor starters. Most relays are manufactured to operate quickly. In a low-voltage application this reduces noise; in a high voltage or current application it reduces [arcing](https://en.wikipedia.org/wiki/Relay#Undesired_arcing).

When the coil is energized with [direct current](https://en.wikipedia.org/wiki/Direct_current), a [diode](https://en.wikipedia.org/wiki/Diode) is often placed across the coil to dissipate the energy from the collapsing magnetic field at deactivation, which would otherwise generate a [voltage spike](https://en.wikipedia.org/wiki/Voltage_spike) dangerous to [semiconductor](https://en.wikipedia.org/wiki/Semiconductor) circuit components. Such diodes were not widely used before the application of [transistors](https://en.wikipedia.org/wiki/Transistor) as relay drivers, but soon became ubiquitous as early [germanium transistors](https://en.wikipedia.org/wiki/Bipolar_junction_transistor#Germanium_transistors) were easily destroyed by this surge. Some automotive relays include a diode inside the relay case.

If the relay is driving a large, or especially a [reactive](https://en.wikipedia.org/wiki/Electrical_reactance) load, there may be a similar problem of surge currents around the relay output contacts. In this case a [snubber](https://en.wikipedia.org/wiki/Snubber) circuit (a capacitor and resistor in series) across the contacts may absorb the surge. Suitably rated capacitors and the associated resistor are sold as a single packaged component for this commonplace use.

If the coil is designed to be energized with [alternating current](https://en.wikipedia.org/wiki/Alternating_current) (AC), some method is used to split the flux into two out-of-phase components which add together, increasing the minimum pull on the armature during the AC cycle. Typically this is done with a small copper "shading ring" crimped around a portion of the core that creates the delayed, out-of-phase component, which holds the contacts during the zero crossings of the control voltage.

**4.3.2 POLE AND THROW OF RELAYS**

Since relays are [switches](https://en.wikipedia.org/wiki/Switch), the terminology applied to switches is also applied to relays; a relay switches one or more *poles*, each of whose contacts can be *thrown* by energizing the coil.

Normally open (NO) contacts connect the circuit when the relay is activated; the circuit is disconnected when the relay is inactive. It is also called a "Form A" contact or "make" contact. NO contacts may also be distinguished as "early-make" or "NOEM", which means that the contacts close before the button or switch is fully engaged.

Normally closed (NC) contacts disconnect the circuit when the relay is activated; the circuit is connected when the relay is inactive. It is also called a "Form B" contact or "break" contact. NC contacts may also be distinguished as "late-break" or "NCLB", which means that the contacts stay closed until the button or switch is fully disengaged.

Change-over (CO), or double-throw (DT), contacts control two circuits: one normally open contact and one normally closed contact with a common terminal. It is also called a "Form C" contact or "transfer" contact ("break before make"). If this type of contact has a "make before break" action, then it is called a "Form D" contact.

The following designations are commonly encountered:

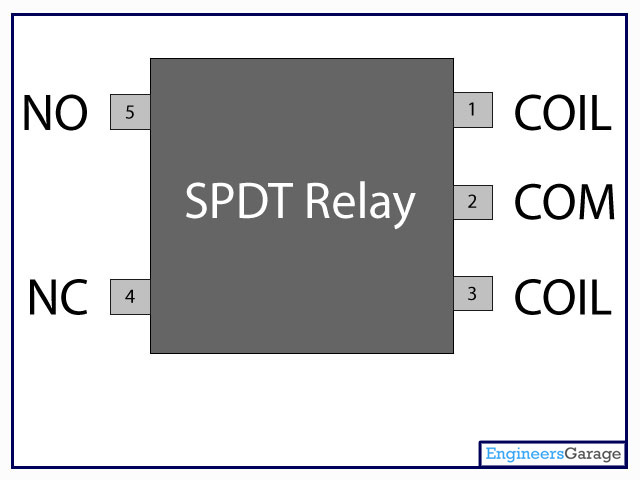
* **SPST** – Single Pole Single Throw. These have two terminals which can be connected or disconnected. Including two for the coil, such a relay has four terminals in total. It is ambiguous whether the pole is normally open or normally closed. The terminology "SPNO" and "SPNC" is sometimes used to resolve the ambiguity.
* **SPDT** – Single Pole Double Throw. A common terminal connects to either of two others. Including two for the coil, such a relay has five terminals in total.
* **DPST** – Double Pole Single Throw. These have two pairs of terminals. Equivalent to two SPST switches or relays actuated by a single coil. Including two for the coil, such a relay has six terminals in total. The poles may be Form A or Form B (or one of each).
* **DPDT** – Double Pole Double Throw. These have two rows of change-over terminals. Equivalent to two SPDT switches or relays actuated by a single coil. Such a relay has eight terminals, including the coil.

**4.3.3 APPLICATIONS**

Relays are used wherever it is necessary to control a high power or high voltage circuit with a low power circuit, especially when galvanic isolation is desirable. The first application of relays was in long [telegraph](https://en.wikipedia.org/wiki/Electric_telegraph) lines, where the weak signal received at an intermediate station could control a contact, regenerating the signal for further transmission. High-voltage or high-current devices can be controlled with small, low voltage wiring and pilots switches. Operators can be isolated from the high voltage circuit. Low power devices such as [microprocessors](https://en.wikipedia.org/wiki/Microprocessor) can drive relays to control electrical loads beyond their direct drive capability. In an automobile, a starter relay allows the high current of the cranking motor to be controlled with small wiring and contacts in the ignition key.

Electromechanical switching systems including [Strowger](https://en.wikipedia.org/wiki/Strowger_switch" \o "Strowger switch) and [Crossbar](https://en.wikipedia.org/wiki/Crossbar_Switch) telephone exchanges made extensive use of relays in ancillary control circuits. The Relay Automatic Telephone Company also manufactured telephone exchanges based solely on relay switching techniques designed by [Gotthilf Ansgarius Betulander](https://sv.wikipedia.org/wiki/Gotthilf_Betulander" \o "sv:Gotthilf Betulander). The first public relay based telephone exchange in the [UK](https://en.wikipedia.org/wiki/UK) was installed in [Fleetwood](https://en.wikipedia.org/wiki/Fleetwood) on 15 July 1922 and remained in service until 1959.

The use of relays for the logical control of complex switching systems like telephone exchanges was studied by [Claude Shannon](https://en.wikipedia.org/wiki/Claude_Shannon), who formalized the application of [Boolean algebra](https://en.wikipedia.org/wiki/Boolean_algebra) to relay circuit design in [A Symbolic Analysis of Relay and Switching Circuits](https://en.wikipedia.org/wiki/A_Symbolic_Analysis_of_Relay_and_Switching_Circuits). Relays can perform the basic operations of Boolean combinatorial logic. For example, the boolean AND function is realised by connecting normally open relay contacts in series, the OR function by connecting normally open contacts in parallel. Inversion of a logical input can be done with a normally closed contact. Relays were used for control of automated systems for machine tools and production lines. The [Ladder programming language](https://en.wikipedia.org/wiki/Ladder_programming_language) is often used for designing [relay logic](https://en.wikipedia.org/wiki/Relay_logic) networks.



**4.4 INTRODUCTION TO PEIZO ELECTRIC TRANSDUCER**

A **piezoelectric transducer** (also known as a piezoelectric sensor) is a device that uses the piezoelectric effect to measure changes in acceleration, pressure, strain, temperature or force by converting this energy into an electrical charge.

A [transducer](https://www.electrical4u.com/transducer-types-of-transducer/) can be anything that converts one form of energy to another. The piezoelectric **material** is one kind of transducers. When we squeeze this piezoelectric material or apply any force or pressure, the transducer converts this energy into voltage. This [voltage](https://www.electrical4u.com/voltage-or-electric-potential-difference/) is a function of the force or pressure applied to it.

The electric voltage produced by a piezoelectric transducer can be easily measured by the voltage [measuring instruments](https://www.electrical4u.com/electrical-measuring-instruments-types-accuracy-precision-resolution-speed/). Since this voltage will be a function of the force or pressure applied to it, we can infer what the force/pressure was by the voltage reading. In this way, physical quantities like mechanical stress or force can be measured directly by using a piezoelectric transducer.



**4.5 INTRUCTION TO RFID MODULE**

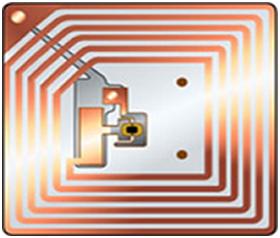
RFID or Radio Frequency Identification System is a technology based identification system which helps identifying objects just through the tags attached to them, without requiring any light of sight between the tags and the tag reader. All that is needed is radio communication between the tag and the reader.



**4.5.1 A BASIC RFID SYSTEM**

**3 Main Components of a RFID System**

* **A RFID tag:** It consists of a silicon microchip attached to a small antenna and mounted on a substrate and encapsulated in different materials like plastic or glass veil and with an adhesive on the back side to be attached to objects.

[](https://www.elprocus.com/wp-content/uploads/2013/09/RFID-Tag.jpg)RFID Tag

* **A reader:** It consists of a scanner with antennas to transmit and receive signals and is responsible for communication with the tag and receives the information from the tag.

[](https://www.elprocus.com/wp-content/uploads/2013/09/An-RFID-reader.jpg)

* **A Processor or a Controller**: It can be a host computer with a Microprocessor or a microcontroller which receives the reader input and process the data.

**4.5.2 TYPES OF RFID SYSTEMS**

* **Active RFID system:**These aresystems where the tag has its own power source like any external power supply unit or a battery. The only constraint being the life time of the power devices. These systems can be used for larger distances and to track high value goods like vehicles.
* **Passive RFID system:**These are systems where the tag gets power through the transfer of power from a reader antenna to the tag antenna. They are used for short range transmission.

Here we are mostly concerned with the passive RFID system as it is most widely used in regular applications like in retail market organizations.

* **A Passive RFID system using Induction coupling method:**In this approach the RFID tag gets power from the reader through inductive coupling method. The reader consists of a coil connected to an AC supply such that a magnetic field is formed around it. The tag coil is placed in the vicinity of the reader coil and an electromotive force is induced it by the virtue of Faraday’s law of induction. The EMF causes a flow of current in the coil, thus producing a magnetic field around it. By the virtue of Lenz law, the magnetic field of the tag coil opposes the reader’s magnetic field and there will be a subsequent increase in the current through the reader coil. The reader intercepts this as the load information. This system is suitable for very short distance communication. The AC voltage appearing across the tag coil is converted to DC using rectifier and filter arrangement.

**4.5.3 WORKING PRINCIPLE**

The whole system uses the passive RFID system with inductive coupling method. As the RFID card (tag) is swiped against the RFID reader, a carrier signal of 125 KHz is send to the tag coil, which receives this signal and modulates them. This modulated signal is received by the reader, interfaced to the microcontroller. The microcontroller receives this data and is programmed to compare it with the data in the existing database. If the data matches, the relevant details of that particular person is displayed on the LCD interfaced to the microcontroller.

**4.6 INTRODUCTION TO 18650 BATTERY**



A **lithium-ion battery** or **Li-ion battery** is a type of [rechargeable battery](https://en.wikipedia.org/wiki/Rechargeable_battery) in which [lithium](https://en.wikipedia.org/wiki/Lithium) [ions](https://en.wikipedia.org/wiki/Ion) move from the negative [electrode](https://en.wikipedia.org/wiki/Electrode) through an [electrolyte](https://en.wikipedia.org/wiki/Electrolyte) to the positive electrode during discharge, and back when charging. Li-ion batteries use an [intercalated](https://en.wikipedia.org/wiki/Intercalation_(chemistry)) lithium [compound](https://en.wikipedia.org/wiki/Chemical_compound) as the material at the positive electrode and typically [graphite](https://en.wikipedia.org/wiki/Graphite) at the negative electrode.

Li-ion batteries have a high [energy density](https://en.wikipedia.org/wiki/Energy_density), no [memory effect](https://en.wikipedia.org/wiki/Memory_effect) (other than [LFP cells](https://en.wikipedia.org/wiki/Lithium_iron_phosphate_battery))[[9]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-9) and low [self-discharge](https://en.wikipedia.org/wiki/Self-discharge). Cells can be manufactured to either prioritize energy or power density. [[10]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-10) They can however be a safety hazard since they contain flammable electrolytes, and if damaged or incorrectly charged can lead to explosions and fires. [Samsung](https://en.wikipedia.org/wiki/Samsung) was forced to recall [Galaxy Note 7](https://en.wikipedia.org/wiki/Samsung_Galaxy_Note_7#Battery_faults) handsets following lithium-ion fires,[[11]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-s7NYT-11) and there have been several incidents involving batteries on [Boeing 787s](https://en.wikipedia.org/wiki/Boeing_787_Dreamliner_battery_problems).

A prototype Li-ion battery was developed by [Akira Yoshino](https://en.wikipedia.org/wiki/Akira_Yoshino) in 1985, based on earlier research by [John Goodenough](https://en.wikipedia.org/wiki/John_Goodenough), [M. Stanley Whittingham](https://en.wikipedia.org/wiki/M._Stanley_Whittingham), [Rachid Yazami](https://en.wikipedia.org/wiki/Rachid_Yazami) and [Koichi Mizushima](https://en.wikipedia.org/wiki/Koichi_Mizushima_(scientist)) during the 1970s–1980s,[[12]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-ieee-12)[[13]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-nobel-13)[[14]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-NIMS-14) and then a commercial Li-ion battery was developed by a [Sony](https://en.wikipedia.org/wiki/Sony) and [Asahi Kasei](https://en.wikipedia.org/wiki/Asahi_Kasei) team led by Yoshio Nishi in 1991.[[15]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-NAE-15)

Lithium-ion batteries are commonly used for [portable electronics](https://en.wikipedia.org/wiki/Portable_electronics) and [electric vehicles](https://en.wikipedia.org/wiki/Electric_vehicle) and are growing in popularity for military and [aerospace](https://en.wikipedia.org/wiki/Aerospace) applications.[[16]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-16)

Chemistry, performance, cost and safety characteristics vary across types of lithium-ion batteries. Handheld electronics mostly use [lithium polymer batteries](https://en.wikipedia.org/wiki/Lithium_polymer_battery) (with a polymer gel as electrolyte), a [lithium cobalt oxide](https://en.wikipedia.org/wiki/Lithium_cobalt_oxide) (LiCoO  
3-based lithium rich layered materials, LMR-NMC), and [lithium nickel manganese cobalt oxide](https://en.wikipedia.org/wiki/Lithium_nickel_manganese_cobalt_oxide) (LiNiMnCoO  
2 or NMC) may offer longer lives and may have better rate capability. Such batteries are widely used for electric tools, medical equipment, and other roles. NMC and its derivatives are widely used in electric vehicles.

Research areas for lithium-ion batteries include extending lifetime, increasing energy density, improving safety, reducing cost, and increasing charging speed,[[19]](https://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-19) among others. Research has been under way in the area of non-flammable electrolytes as a pathway to increased safety based on the flammability and volatility of the organic solvents used in the typical electrolyte. Strategies include [aqueous lithium-ion batteries](https://en.wikipedia.org/wiki/Aqueous_lithium-ion_battery), ceramic solid electrolytes, polymer electrolytes, ionic liquids, and heavily fluorinated systems

**CHAPTER 5**

**PROGRAM CODE**

#include <LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

int i = 0;

char personid[12];

int count = 0;

void setup() {

pinMode(A0, OUTPUT);

pinMode(A5, INPUT);

pinMode(13, OUTPUT);

digitalWrite(A0, HIGH);

Serial.begin(9600);

lcd.begin(16, 2);

lcd.setCursor(0, 0);

lcd.print("RFID MOBILE ");

lcd.setCursor(0, 1);

lcd.print("CHARGING ");

delay(3000);

lcd.clear();

}

void loop() {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("SCAN THE CARD ");

lcd.setCursor(0, 1);

lcd.print("CHARGE UR PHONE ");

if (Serial.available() > 0)

{

for (i = 0; i < 12; i++)

personid[i] = Serial.read();

personid[i] = '\0';

if (strncmp(personid, "0B00269BBC0A", 12) == 0)

{

digitalWrite(A0, LOW);

for (i = 60; i >= 0; i--)

{

if (analogRead(A5) > 0)

digitalWrite(13, HIGH);

else

digitalWrite(13, LOW);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("MANISHA ");

lcd.setCursor(0, 1);

lcd.print(i);

delay(1000);

}

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("TIME OUT ");

digitalWrite(A0, HIGH);

delay(2000);

}

else if (strncmp(personid, "0B0026912A96", 12) == 0)

{

digitalWrite(A0, LOW);

for (i = 60; i >= 0; i--)

{

if (analogRead(A5) > 0)

digitalWrite(13, HIGH);

else

digitalWrite(13, LOW);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("THAFEEM ");

lcd.setCursor(0, 1);

lcd.print(i);

delay(1000);

}

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("TIME OUT ");

digitalWrite(A0, HIGH);

delay(2000);

}

else if (strncmp(personid, "0B002690CE73", 12) == 0)

{

digitalWrite(A0, LOW);

for (i = 60; i >= 0; i--)

{

if (analogRead(A5) > 0)

digitalWrite(13, HIGH);

else

digitalWrite(13, LOW);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("JASHWANTH ");

lcd.setCursor(0, 1);

lcd.print(i);

delay(1000);

}

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("TIME OUT ");

digitalWrite(A0, HIGH);

delay(2000);

}

else if (strncmp(personid, "0B0026C857B2", 12) == 0)

{

digitalWrite(A0, LOW);

for (i = 60; i >= 0; i--)

{

if (analogRead(A5) > 0)

digitalWrite(13, HIGH);

else

digitalWrite(13, LOW);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("NAGARAJU ");

lcd.setCursor(0, 1);

lcd.print(i);

delay(1000);

}

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("TIME OUT ");

digitalWrite(A0, HIGH);

delay(2000);

}

}

if (analogRead(A5) > 0)

digitalWrite(13, HIGH);

else

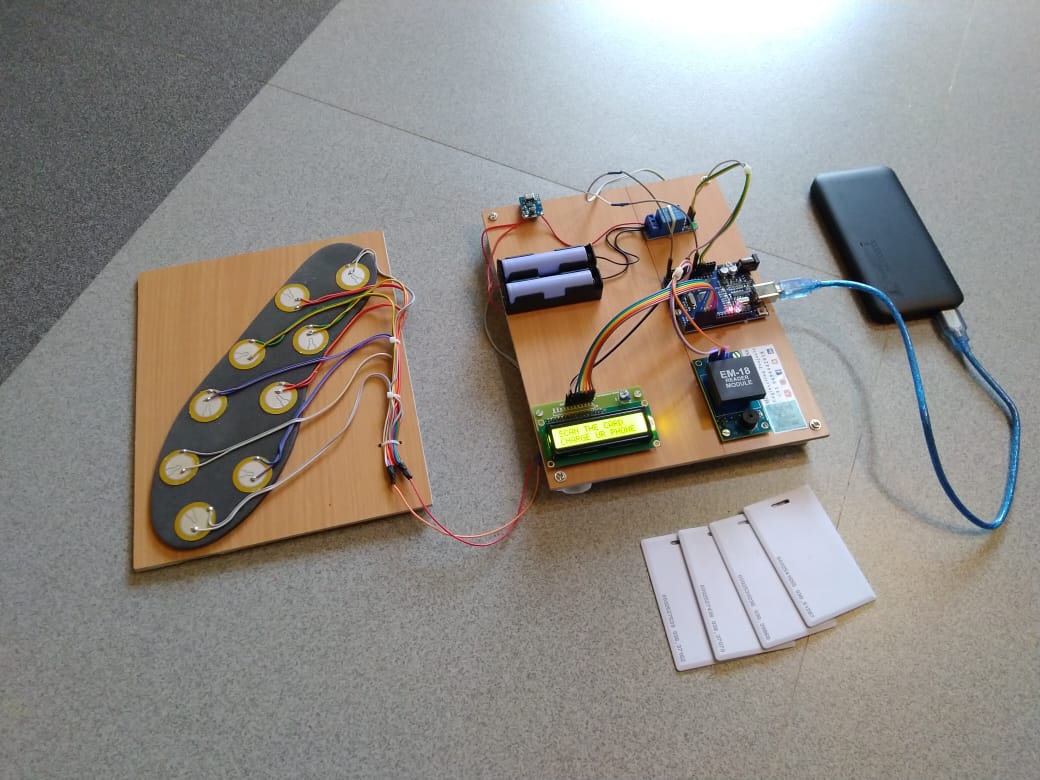
digitalWrite(13, LOW);

delay(100);

}

**CHAPTER 6**

**RESULT OF THE PROJECT**

****

**CHAPTER 7**

**CONCLUSION**

Day by day, the population of the country is increasing and the requirement of the power is also increasing. At the same time the wastage of energy is also increasing in many ways. So, reforming this energy back to a usable form is the major solution. In this footstep power generation project, we are generating power with the help of human’s footsteps; this power is then used to charge battery. The power is stored in a battery that can be used to charge a mobile phone using RFID card. This system is powered by Atmega 328 microcontroller, it consists of Arduino IDE, RFID sensor, USB cable and LCD. When we power on the system, the system enters into registration mode. We can register three users. Once all the user is entered in the system then the system asks to swipe the card and connect the charger. Initially all the user is given 5 minutes of charging time as default.

**BIBLIOGRAPHY**

Ahmad, N.A.N. and Zamri, S.A.S., 2014, September. The cross platform application development adapted Spring framework to support front-end tendering services. In Computer, Communications, and Control Technology (I4CT), 2014 International Conference on (pp. 58-62). IEEE.

Alarcon, R. and Wilde, E., 2010, April. Linking data from restful services. In Third Workshop on Linked Data on the Web, Raleigh, North Carolina (April 2010).

Al-Fagih, A.E., Al-Turjman, F.M., Alsalih, W.M. and Hassanein, H.S., 2013. A priced public sensing framework for heterogeneous IoT architectures. Emerging Topics in Computing, IEEE Transactions on, 1(1), pp.133-147.

Alpaydin, E., 2014. Introduction to machine learning. MIT press.

Armando, A., Carbone, R., Chekole, E.G. and Ranise, S., 2014, June. Attribute based access control for APIs in spring security. In Proceedings of the 19th ACM symposium on Access control models and technologies (pp. 85-88). ACM.

Atighetchi, M., Soule, N., Pal, P., Loyall, J., Sinclair, A. and Grant, R., 2013, October. Safe configuration of TLS connections. In Communications and Network Security (CNS), 2013 IEEE Conference on (pp. 415- 422). IEEE.

Bormann, C., Castellani, A.P. and Shelby, Z., 2012. Coap: An application protocol for billions of tiny internet nodes. IEEE Internet Computing, 16(2), p.62.

Centenaro, M., Vangelista, L., Zanella, A. and Zorzi, M., 2015. Long-Range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios. arXiv preprint arXiv:1510.00620.

Chinrungrueng, J., Sunantachaikul, U. and Triamlumlerd, S., 2007, January. Smart parking: An application of optical wireless sensor network. In Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on (pp. 66-66). IEEE.