Ministry of Higher Education
Faculty of Engineering Mansoura University
Communication & information Engineering Department

Graduation Project

Auto-controlled protection system

Project Team:

- **Hader ezz-Elarb Mustafa.**
- **Salma Mustafa Ellaithy.**
- **Noran Mohamed Gad.**
- **Khaled Mohamed Ali.**
- **Hisham Abd-Ellatif Abd-Ellatif.**
- **Mohamed Khaled Abd-Ellhalem.**
- **Ahmed Nagy Mohamed.**
- **Ahmed Karim Abd-Elalem.**
- **Mohamed Samy Ahmed Mohamed.**

Supervisors:

- **Assoc.Prof. : Sherif Hussein.**
- **Eng. Rana Mohamed.**

Acknowledgements:

All thanks and appreciation are attributed to **Assoc.pro. Sherif Hussien** and **Dr. Mohamed Sabry Saraya** who not only served as our supervisor but also encouraged and challenged us throughout our academic program. **Eng. Rana Mohamed**, and all other department staff guided us through the learning process , never accepting less than our best efforts. We thank them all.

Content list

2- Introduction

Now in the twenty-first century after the great technological development, man was able to translate everything in our daily life from mere ideas to tangible things on the ground, which led to an engineering and medical breakthrough in the world, and everything became smooth, simple, and somewhat error free. Also to make things simple and controllable in all areas.

In light of the scientific development, there are engineers who are able to make the difference and determine the fate of some things in the world. They were also able to make small things that control large parts as we see daily in our lives, as they are now able to reach the moon.

They are now able to achieve dreams with this science, but there are som difficulties and obstacles for these engineers, including the ignorance of some countries with this science, including poverty, which may lead to the non-use of all of these sciences.

But with knowledge, poverty can be eradicated, making the world free from ignorance, more light and more developed.

Communications and information engineers were able to develop scientific in most of the engineering and non-engineering fields made of things very complicated after we thought that they are impossible simple and more realistic things where they dealt with technology in everything and made it an important element but rather an element of life on which many inventions are based, but all the inventions in Our lives where they were able to intervene also in the medical, management, marketing, industrial, and licensing fields, and control some projects in many commercial markets and make these goods produced by them control and control the income of many countries and opened the doors Soft domestic and international cooperation through the import and export, which made these things are important, but when many people appear to be the source of their daily lives.

But despite all this development and all this science and technology, there are also those who use this science in a lot of harmful things that are destroyed, not built, and there are also those who use it to demolish people and spread ignorance and stupidity in many generations to come, especially in late countries or countries of the second world as it is called Accordingly, the purpose of this is to create generations that submit to orders only, which are difficult for them to make decisions in their lives in order

to raise their country and put it in the best form and in the best condition.
Communications and information engineers were able to invent and develop many of the things that we use in our daily life and now it is indispensable for these things, for example (the mobile phone that they were able to develop daily after day and became one of the most important things in life and also many tablet devices.

And sought intervention in the field of industry, such as automobile industry, and there are now cars that move faster and also can move without a driver.

Also in the field of protection, it has become possible to manage many important personal, governmental or international interests by coordinating some systems and using them by these engineers)

One of these systems is this system or this project.

(Auto-controlled protection system)

Which is one of the most important projects used for embeeded systems engineers.

Because it is used to facilitate some of the tasks found in homes or government departments and in this project we will use it in any protection system used in the process of protecting a bank by using some tools and sensors and some programming to form a special system and security difficult to penetrate

In our **smart Bank** At this part of this project we used some features to support our security system idea , like :

1- Arduino mega 2056. 2- PIR
Sensor (motion system) 3- LDR
Sensor & laser.

4- MQ2 Sensor (gas and smoke).
5- DHT11 Sensor (Temperature and Humidity).
6- Design of maquette.

And we use this sensor for :

***MQ2:** And we use it to control the smoke and gas in place and if the find smoke or high at gas level make alarm to me .

***PIR:** And we use this sensor to Calculate the number of objects passing through the doors and appear on the 7-segment to counter.

***DHT11**: We use this sensor at project to calculate the temp. & hum With Lcd (16*2) and potentiometer to show degree In the place.

***LDR:** We use it with “buzzer” to identify foreign objects and alert them as they pass

Next project Phase:-

1- . Adding Door locker connected to keypad to employee room.

2- Adding 7 Segment counter to count no of cars entering/outing the garage.

3- Adding more PIR sensors and LDR sensors.

Chapter 1

Arduino

INTRODUCTION OF ARDUINO:

I wanted to begin by thanking Element14 and the sponsors for selecting me to participate in the Arduino Mega 2560 Road Test.
I am really excited to have been chosen to Road Test the Arduino platform, which has dramatically changed the landscape of microcontroller electronics.
The user base has really exploded recently and it seems like I've been hearing about it more and more lately, even from major publications (Make Magazine, Wired, New York Times).
Most Element14 members/readers are probably already familiar with the Arduino IDE and boards, but if not I will try to sum it up in as few words as possible:
Standardized open-source hardware microcontroller platform based on Atmel ATmega 8-bit devices and a cross-platform open-source software integrated design environment. In other words, Arduino provides artists, engineers, hobbyists, makers, etc. with an inexpensive community-driven programmable electronics platform with which they can rapidly prototype or realize their designs or use as a basis for learning hardware and software engineering.

BACKGROUND:

I have been an Arduino user for about 3 years now, but this is my first opportunity to test out the Mega version of the platform.
In the past I have used the Diecimila, Duemilanove, Pro Mini, and the Uno, as well as creating my own Arduino-compatible PCBs.
The integrated design environment (IDE) software has evolved through several revisions since then and it now provides many great built-in libraries and functions.
I already had the latest version of the IDE installed prior to receiving the Mega for review, so I was up and running immediatel

UNBOXING:

The Mega 2560 comes in their new (as of the release of the Uno, previously just a static bag) simple packaging: nicely printed card-stock box folded around the board inside an ant-static bag.
Upon opening the box you find an insert that thanks the user for supporting Arduino (see picture below).
It provides information about how/where the Arduino is built and a warranty statement.
Also included is a small sheet of stickers if you want to advertise your project as having "Arduino Inside." Since there are no jumpers to set or any switches on the board, the next step is to just plug it in.

An Arduino board is a one type of microcontroller based kit. The first Arduino technology was developed in the year 2005 by David Cuartielles and Massimo Banzi.
The designers thought to provide easy and low cost board for students, hobbyists and professionals to build devices. Arduino board can be purchased from the seller or directly we can make at home using various basic components.
The best examples of Arduino for beginners and hobbyists includes motor detectors and thermostats, and simple robots. In the year 2011, Adafruit industries expected that over 3lakhs Arduino boards had been produced.
But, 7lakhs boards were in user’s hands in the year 2013.
Arduino technology is used in many operating devices like communication or controlling.

Arduino "history":

is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

Its products are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL) permitting the manufacture of Arduino boards and software distribution by anyone.

Arduino boards are available commercially in preassembled form or as doit-yourself (DIY) kits.

Arduino board designs use a variety of microprocessors and controllers.

The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (For prototyping) and other circuits.

The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers.

The microcontrollers can be programmed using C and C++ programming languages.

In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

The Arduino project started in 2005 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a lowcost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats and motion detectors.

The name *Arduino* comes from a bar in Ivrea, Italy, where some of the founders of the project used to meet.

The bar was named after Arduino of Ivrea, who was the margrave of the March of Ivrea and King of Italy from 1002 to 1014.

In early 2008, the five co-founders of the Arduino project created a company, Arduino LLC, to hold the trademarks associated with Arduino. The manufacture and sale of the boards was to be done by external companies, and Arduino LLC would get a royalty from them.

The founding bylaws of Arduino LLC specified that each of the five founders transfer ownership of the Arduino brand to the newly formed company.

At the end of 2008, Gianluca Martino's company, Smart Projects, registered the Arduino trademark in Italy and kept this a secret from the other co-founders for about two years.

This was revealed when the Arduino company tried to register the trademark in other areas of the world (they originally registered only in the US), and discovered that it was already registered in Italy.

Negotiations with Gianluca and his firm to bring the trademark under control of the original Arduino company failed.

In 2014, Smart Projects began refusing to pay royalties.

They then appointed a new CEO, Federico Musto, who renamed the company *Arduino SRL* and created the website *arduino.org*, copying the graphics and layout of the original *arduino.cc*.

This resulted in a rift in the Arduino development team.

In January 2015, Arduino LLC filed a lawsuit against Arduino SRL.

In May 2015, Arduino LLC created the worldwide trademark **Genuino**, used as brand name outside the United States.

At the World Maker Faire in New York on 1 October 2016, Arduino LLC cofounder and CEO Massimo Banzi and Arduino SRL CEO Federico Musto announced the merger of the two companies.

By 2017 Arduino AG owned many Arduino trademarks. In July 2017 BCMI, founded by Massimo Banzi, David Cuartielles, David Mellis and Tom Igoe, acquired Arduino AG and all the Arduino trademarks.

Fabio Violante is the new CEO replacing Federico Musto, who no longer works for Arduino AG.

Arduino “Hardware”:

Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website.

Layout and production files for some versions of the hardware are also available.

Although the hardware and software designs are freely available under copyleft licenses, the developers have requested the name *Arduino* to be exclusive to the official product and not be used for derived works without permission.

The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product. Several Arduino-compatible products commercially released have avoided the project name by using various names ending in *arduino*.

Most Arduino boards consist of an Atmel 8-bit AVR microcontroller (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560) with varying amounts of flash memory, pins, and features.

The 32-bit Arduino Due, based on the Atmel SAM3X8E was introduced in 2012.

The boards use single or double-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed *shields*.

Multiple and possibly stacked shields may be individually addressable via an I²C serial bus.

Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator or ceramic resonator.

Some designs, such as the LilyPad, run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions.

Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory.

The default bootloader of the Arduino UNO is the optiboot bootloader.

Boards are loaded with program code via a serial connection to another computer.

Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232.

Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header.

Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods.

When used with traditional microcontroller tools, instead of the Arduino IDE,

standard AVR in-system programming (ISP) programming is used.

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits.

The *Diecimila*, *Duemilanove*, and current *Uno* provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins.

These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers.

Several plug-in application shields are also commercially available.

The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

Many Arduino-compatible and Arduino-derived boards exist.

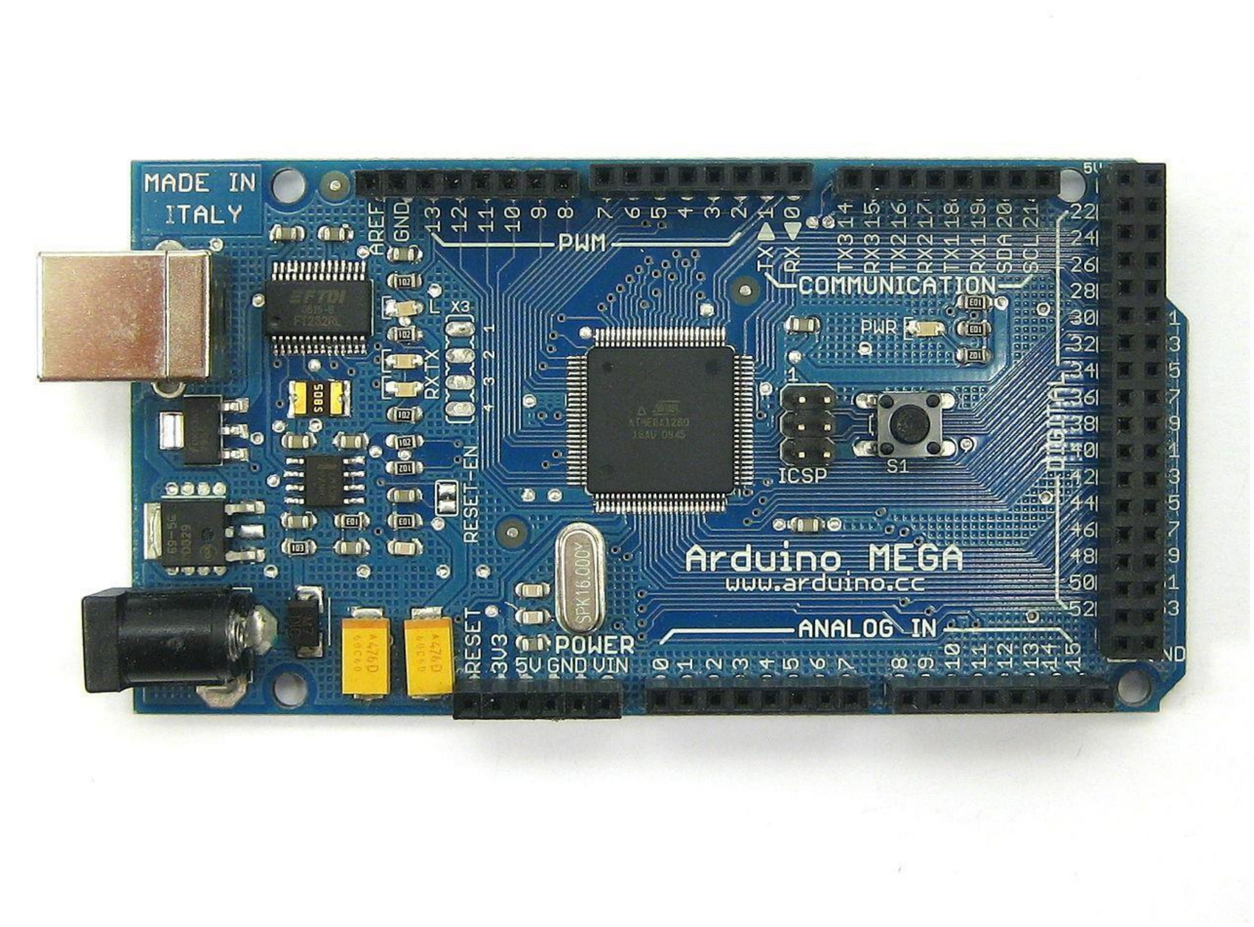
Some are functionally equivalent to an Arduino and can be used interchangeably.

Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots.

Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not.

Some variants use different processors, of varying compatibility.

Official boards:



Further information: List of Arduino boards and compatible systems

The original Arduino hardware was produced by the Italian company Smart Projects. Some Arduino-branded boards have been designed by the

American companies SparkFun Electronics and Adafruit Industries. As of 2016, 17 versions of the Arduino hardware have been commercially produced.

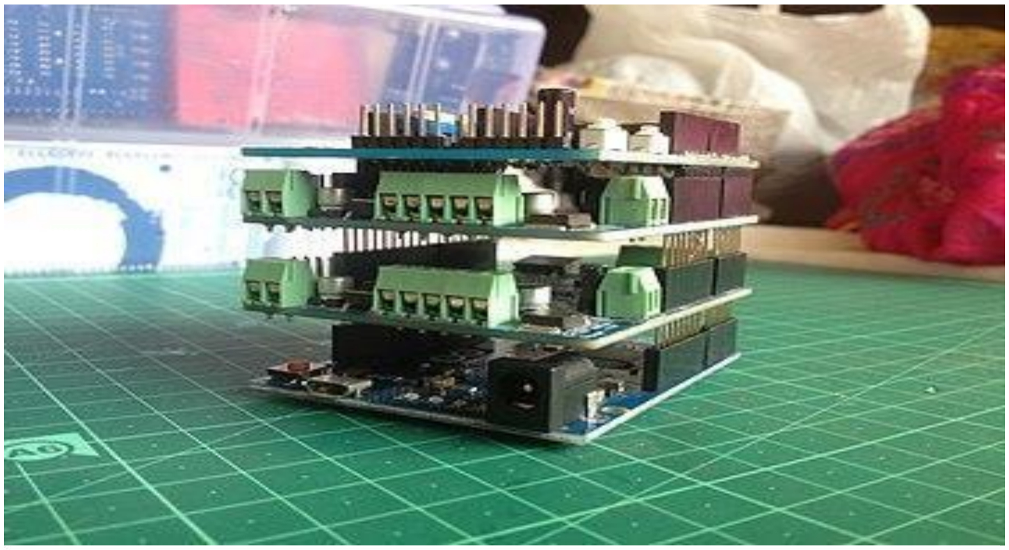
Shields:

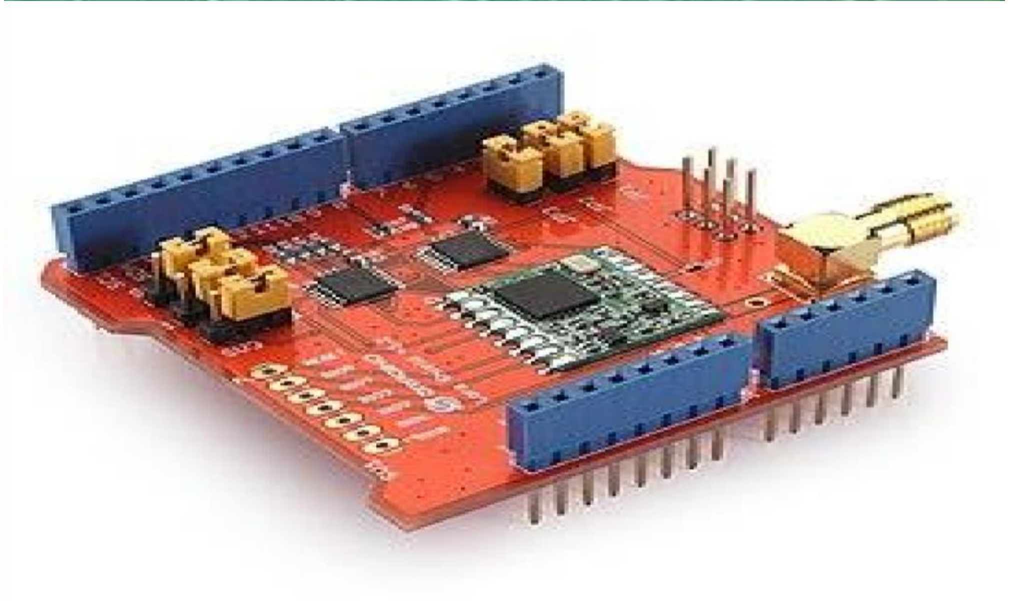
Arduino and Arduino-compatible boards use printed circuit expansion boards called *shields*, which plug into the normally supplied Arduino pin headers.

Shields can provide motor controls for 3D printing and other applications, GNSS (satellite navigation), Ethernet, liquid crystal display (LCD), or breadboarding (prototyping).

Several shields can also be made do it yourself (DIY).

Some of this shiled is





Atmel provides a development environment for their 8-bit AVR and 32bit ARM Cortex-M based microcontrollers: AVR Studio (older) and Atmel Studio (newer).

IDE :

The Arduino integrated development environment (IDE) is a crossplatform application (for Windows, macOS, Linux) that is written in the programming language Java.

It originated from the IDE for the languages *Processing* and *Wiring*.

It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board.

It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

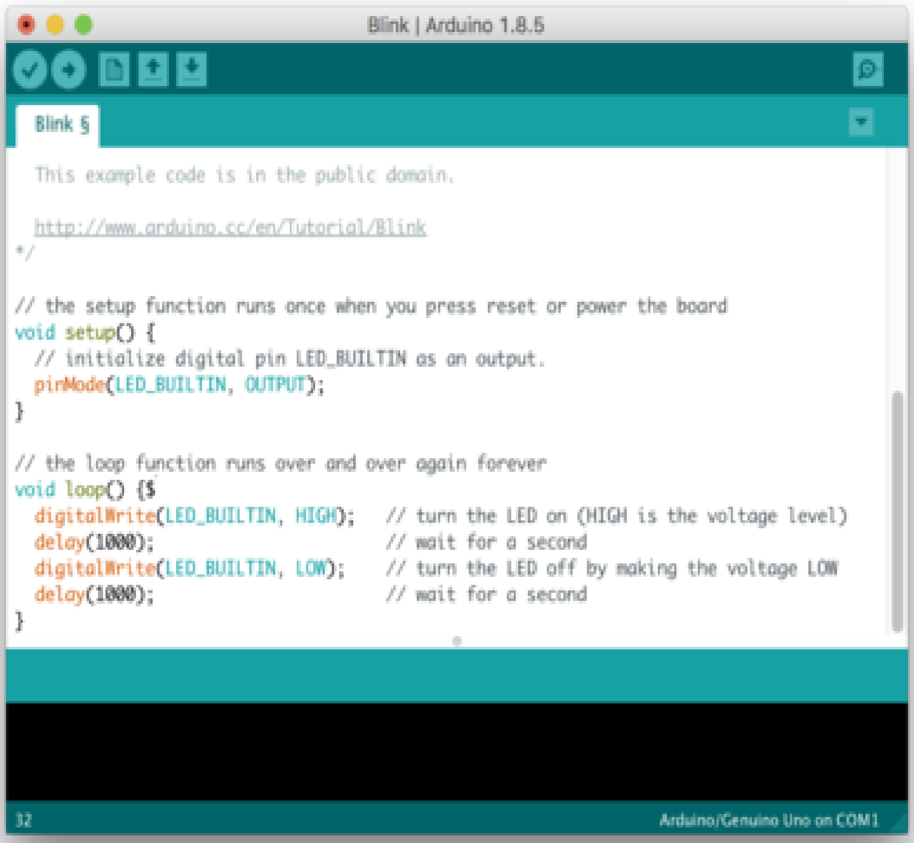
The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures.

User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.

The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

Arduino software IDE:



```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {$
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

32

Pro IDE:

On October 18th, 2019, Arduino Pro IDE (alpha preview) was released. The system still uses Arduino CLI (Command Line Interface), but improvements include a more professional development environment, autocompletion support, and Git integration .

The application frontend is based on the Eclipse Theia Open Source IDE. The main features available in the alpha release are:

- Modern, fully featured development environment.
- Dual Mode, Classic Mode (identical to the Classic Arduino IDE) and Pro Mode (File System view).
- New Board Manager.
- New Library Manager.
- Board List.
- Basic Auto-Completion. (Arm targets only) Git Integration.
- Serial Monitor. Dark Mode.

Sketch:

A *sketch* is a program written with the Arduino IDE.[59] 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 powerup 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(1)`.

Blink example:

Power LED (red) and User LED (green) attached to pin 13 on an arduino compatible board

Most Arduino boards contain a light-emitting diode (LED) and a current limiting resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions. A typical program used by beginners, akin to Hello, World!, is "blink", which repeatedly blinks
the on-board LED integrated into the Arduino board. This program uses the functions `pinMode()`, `digitalWrite()`, and `delay()`, which are provided by the internal libraries included in the IDE environment. This program is usually loaded into a new Arduino board by the manufacturer.

```
#define LED_PIN 13                    // Pin number attached to LED.

void setup() {
    pinMode(LED_PIN, OUTPUT);          // Configure pin 13 to be a digital
output.
}  void loop() {     digitalWrite(LED_PIN, HIGH);    //
Turn on the LED.
    delay(1000);                      // Wait 1 second (1000 milliseconds).
digitalWrite(LED_PIN, LOW);     // Turn off the LED.     delay(1000);
// Wait 1 second.
```

Libraries:

The open-source nature of the Arduino project has facilitated the publication of many free software libraries that other developers use to augment their projects.

Recognitions:

The Arduino project received an honorary mention in the Digital Communities category at the 2006 Prix Ars Electronica.

Applications:

- Arduboy, a handheld game console based on Arduino
- Arduinome, a MIDI controller device that mimics the Monome Ardupilot, drone software and hardware ArduSat, a cubesat based on Arduino.
- C-STEM Studio, a platform for hands-on integrated learning of computing, science, technology, engineering, and mathematics (CSTEM) with robotics.
- Data loggers for scientific research.[67][68][69][70]
- OBDuino, a trip computer that uses the on-board diagnostics interface found in most modern cars
- OpenEVSE an open-source electric vehicle charger

MEGA-SIZED:

The Mega 2560 is significantly larger than its little brother the Uno (4" x 2.1" versus 2.7" x 2.1", ~50% larger), but in this case Mega is a relative term. It is still smaller than an iPhone 4 (4.5" x 2.31"), but it would only be appropriate for projects where you have a lot of room or where you need a lot of I/O. The Mega has 54 digital I/O pins (14 which support PWM using the Atmel hardware timers), compared to 14 (6 PWM) on the Uno. You get more of just about everything, actually - here's a little table for comparison:

	Mega 2560	Uno
Digital I/O	54	14
PWM	14	6
Analog In	16	6
Serial	4	1
I2C	1	1
SPI	1	1
Interrupt Pins	5	2
Flash	256	32
RAM	8	2
EEPROM	4	1

Besides that, all the other specs are the same (voltage, current, USB interface, etc.). The pin numbers and corresponding pin functions from the Uno are mostly the same/in the same place on the Mega 2560, however some things have been moved around (notably I2C). This spreadsheet does an awesome job of summarizing all the pins and functions between the Uno and the Mega 2560.

INITIAL IMPRESSIONS

There are a lot of pins! The Mega 2560 may only be 50%

larger in size than the Uno but it has almost 4 times the I/O, thanks mainly to the large double row at the back.

It's like an stretch limousine version of an Uno. I like how the pins are grouped by special function - PWM, Communication, Digital, Analog, Power

and that the Arduino Team did their best to keep things in the same place so that shields made for the smaller boards would still work on the Mega. Speaking of shields, someone needs to make a mini shield to access the back portion of the PCB

that extends beyond where normal shields stop. I could see it matching up with the outline that most shields have (to access the ISP pins) like jigsaw puzzle pieces.

With its vast array of I/O and additional memory space, the Mega could be very useful for some projects, such as robotics, LED arrays, or sensor interfaces. However, for those just starting out with microcontrollers or Arduino, it's probably overkill. The retail price difference, $50-$65 for the Mega versus $30 for the Uno, would also sway people towards the Uno and limit the usage of the Mega to projects that really need the added features.
Below I have listed some additional thoughts about the Mega

Pros:

- Big - lots of I/O for projects that need it
- Memory - if your code simply won't fit in the Uno, you've got much more room here
- Same/Similar shape/layout - many shields made for the smaller boards will still work
- Nicely organized - I/O grouped by function Cons:
- Big - not for your space-constrained project
- Price - about double the price of the Uno
- Shield Compatibility - limited support for large I/O expansion header

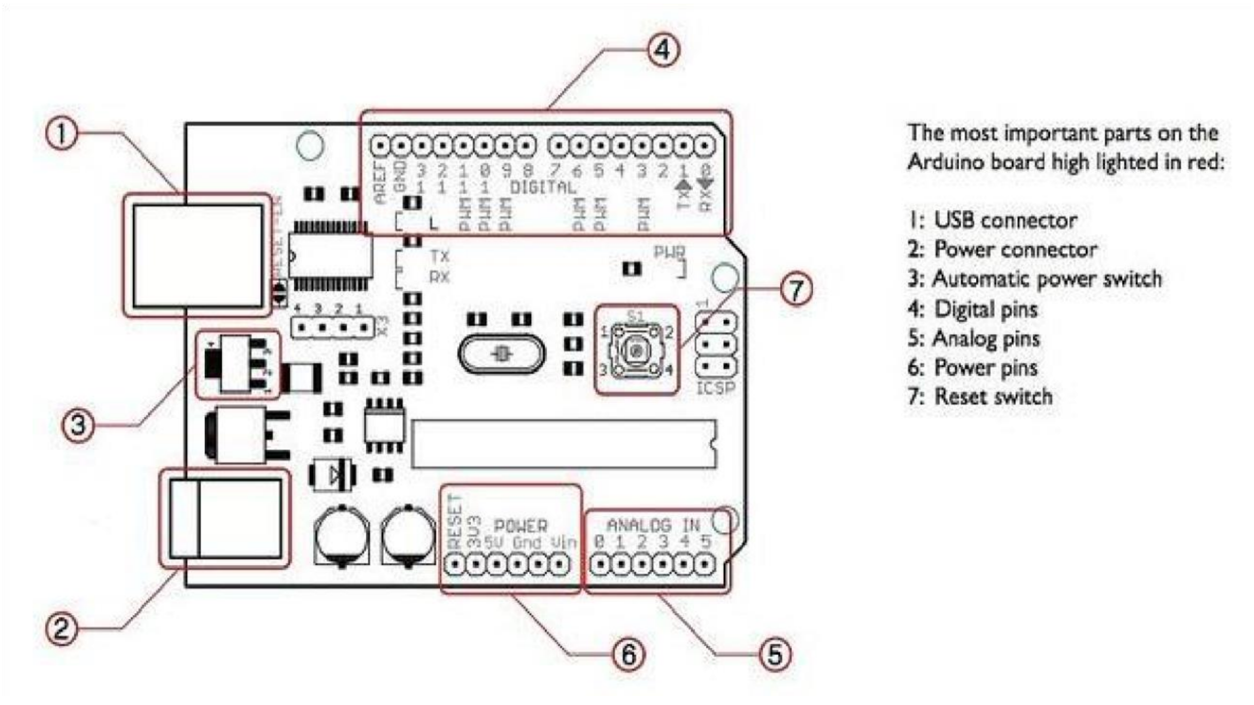
Cons:

- Big - not for your space-constrained project
- Price - about double the price of the Uno
- Shield Compatibility - limited support for large I/O expansion header

The main **advantage** of the **Arduino** technology is, you can directly load the programs into the device without the need of a hardware programmer to burn the program.

Arduino Technology:

A typical example of the Arduino board is Arduino Uno.It includes an ATmega328 microcontroller and it has 28-pins



Arduino Pin Diagram:

The pin configuration of the Arduino Uno board is shown in the above. It consists of 14-digital i/o pins. Wherein 6 pins are used as pulse width modulation o/ps and 6 analog i/ps, a USB connection, a power jack, a 16MHz crystal oscillator, a reset button, and an ICSP header.

Arduino board can be powered either from the personal computer through a USB or external source like a battery or an adaptor.

This board can operate with an external supply of 7-12V by giving voltage reference through the IORef pin or through the pin Vin.

Digital I/Ps

It comprises of 14-digital I/O pins, each pin take up and provides 40mA current. Some of the pins have special functions like pins 0 & 1, which acts as a transmitter and receiver respectively.

For serial communication, pins-2 & 3 are external interrupts, 3,5,6,9,11 pins delivers PWM o/p and pin-13 is used to connect LED.

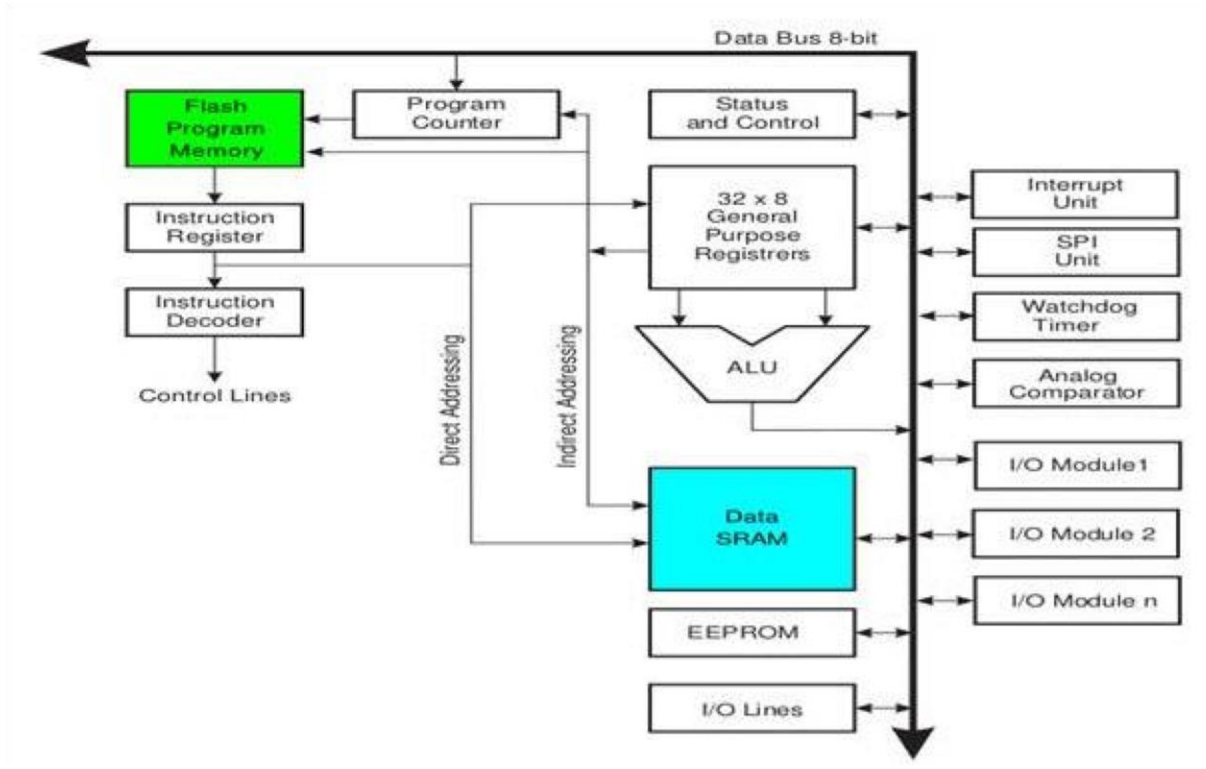
Analog i/ps: It has 6-analog I/O pins, each pin provide a 10 bits resolution.

Aref: This pin gives a reference to the analog i/ps.

Reset: When the pin is low, then it resets the microcontroller.

Arduino Architecture:

Basically, the processor of the Arduino board uses the Harvard architecture where the program code and program data have separate memory. It consists of two memories such as program memory and data memory. Wherein the data is stored in data memory and the code is stored in the flash program memory. The Atmega328 microcontroller has 32kb of flash memory, 2kb of SRAM 1kb of EPROM and operates with a 16MHz clock speed.



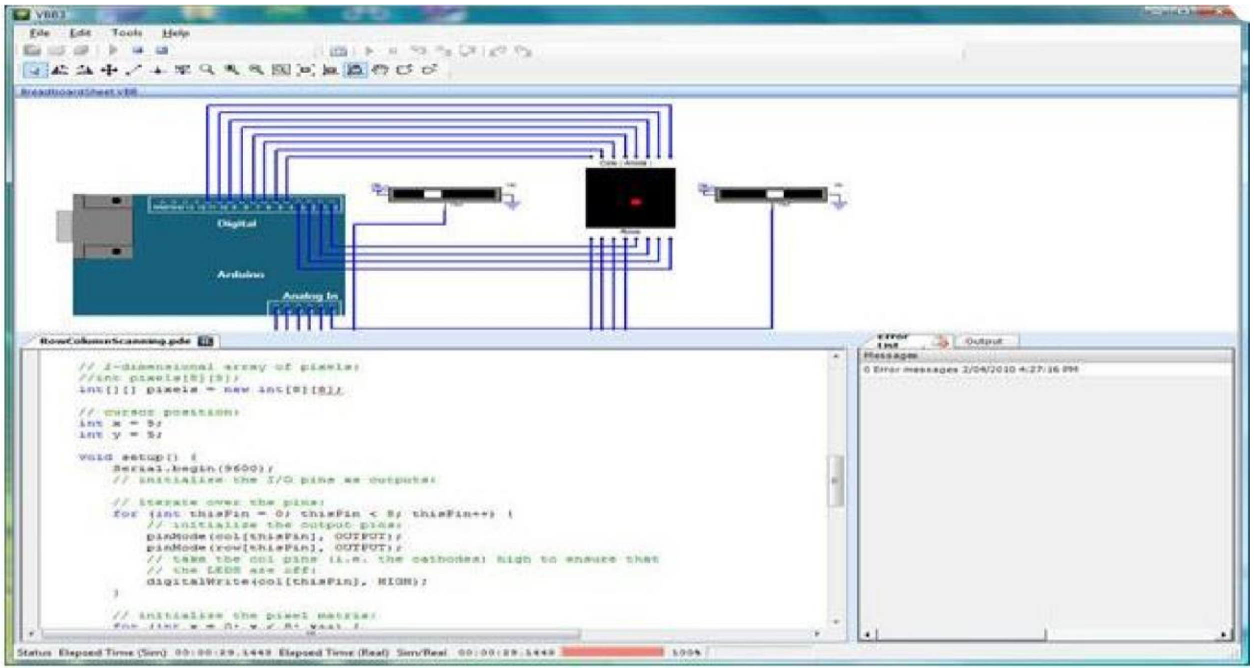
How to program an Arduino?

The main advantage of the Arduino technology is, you can directly load the programs into the device without the need of a hardware programmer to burn the program.

This is done because of the presence of the 0.5KB of boot loader, that allows the program to be dumped into the circuit.

The Arduino tool window contains a toolbar with a various buttons like new, open, verify, upload and serial monitor.

And additionally it comprises of a text editor (employed to write the code), a message space (displays the feedback) like showing the errors, the text console, that displays the o/p & a series of menus just like the file, tool menu & edit.



```
// 2-dimensional array of pixels:
//int pixels[8][8];
int[][] pixels = new int[8][8];

// cursor position:
int x = 5;
int y = 5;

void setup() {
    Serial.begin(9600);
    // initialize the I/O pins as outputs:

    // iterate over the pins:
    for (int thisPin = 0; thisPin < 8; thisPin++) {
        // initialize the output pins:
        pinMode(col[thisPin], OUTPUT);
        pinMode(row[thisPin], OUTPUT);
        // take the col pins (i.e. the cathodes) high to ensure that
        // the LEDS are off:
        digitalWrite(col[thisPin], HIGH);
    }

    // initialize the pixel matrix:
```

```
Message
0 Error messages 2/04/2010 4:27:16 PM
```

```
Status  Elapsed Time (Sim) 00:00:29.1449  Elapsed Time (Real)  Sim/Real  00:00:29.1449  100%
```

- Programming into the Arduino board is called as sketches. Each sketch contains of three parts such as Variables Declaration, Initialization and Control code. Where, Initialization is written in the setup function and Control code is written in the loop function.
- The sketch is saved with .ino and any operation like opening a sketch, verifying and saving can be done using the tool menu.
- The sketch must be stored in the sketchbook directory.
- Select the suitable board from the serial port numbers and tools menu.
- Select the tools menu and click on the upload button, then the boot loader uploads the code on the microcontroller.

The Obstacle Avoidance Robot Operated with Arduino:

The main concept of this project is to design a robot using ultrasonic sensors to avoid the obstacle. A robot is a machine and it is a combination of programs instructions and motors. It can perform some task with some guidance or automatically. This robotic vehicle has an intelligence which is built inside of the robot. When a obstacle problem comes ahead of it then, it guides itself. This robot is designed with a microcontroller from Atmel family of Aduino board.

Basic Functions of Arduino Technology

- Digital read pin reads the digital value of the given pin.
- Digital write pin is used to write the digital value of the given pin.
- Pin mode pin is used to set the pin to I/O mode.
- Analog read pin reads and returns the value.
- Analog write pin writes the value of the pin.
- Serial. Begins pin sets the beginning of serial communication by setting the rate of bit.

Advantages of Arduino Technology:

- It is cheap
- It comes with an open supply hardware feature that permits users to develop their own kit
- The software of the Arduino is well-suited with all kinds of in operation systems like Linux, Windows, and Macintosh, etc.
- It also comes with open supply software system feature that permits tough software system developers to use the Arduino code to merge with the prevailing programing language libraries and may be extended and changed.
- For beginners, it is very simple to use.

Arduino based Home Automation

The main goal of this project is to design a home automation system using an Arduino board with Bluetooth being controlled remotely by any Android OS based smart phone.

This home automation system provides a modern solution with smart phones.In order to achieve this, a Bluetooth device is attached to the Arduino board at the receiver side and while on the transmitter side,

a GUI application on the smart phone sends ON/OFF commands to the receiver where loads are connected.

By touching the particular location on the graphical user interface (GUI), the different loads can be remotely turned ON/OFF via this technology.
When we touch the exact location on the GUI,
then the loads can be turned ON/OFF remotely.
The loads works with an Arduino board through Thyristors and Opto-Isolators using triacs.

Chapter(2)

sensors

(1-2) Temperature and humidity sensor(DHT11):

Introduction:

This DHT11 Temperature and Humidity Sensor features a calibrated digital signal output with the temperature and humidity sensor capability.

It is integrated with a high-performance 8-bit microcontroller.

Its technology ensures the high reliability and excellent long-term stability.

This sensor includes a resistive element and a sensor for wet NTC temperature measuring devices.

It has excellent quality, fast response, anti-interference ability and high performance.

Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber.

The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, we should call these calibration coefficients.

The single-wire serial interface system is integrated to become quick and easy.

Small size, low power, signal transmission distance up to 20 meters, enabling a variety of applications and even the most demanding ones.

The product is 4-pin single row pin package.

Convenient connection, special packages can be provided according to users need.

Feature & Application:

* Full range temperature compensated

* Relative humidity and temperature measurement

* Calibrated digital signal

*Outstanding long-term stability

*Extra components not needed

* Long transmission distance

* Low power consumption

*4 pins packaged and fully interchangeable

DESCRIPTION:

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor.

It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).

Its fairly simple to use, but requires careful timing to grab data.

The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old

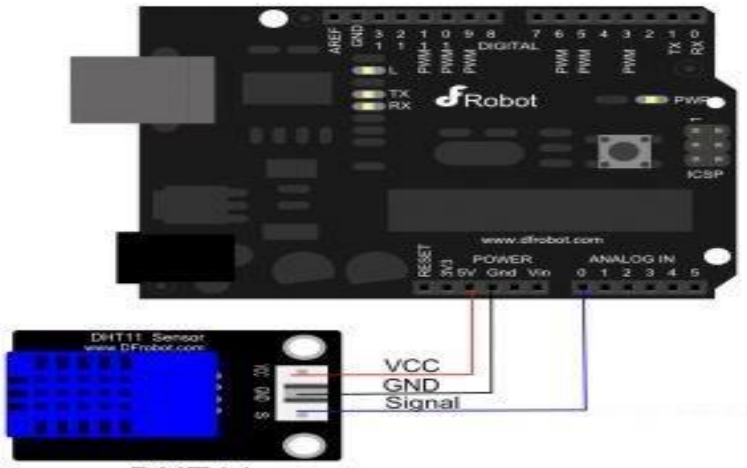
Specification:

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

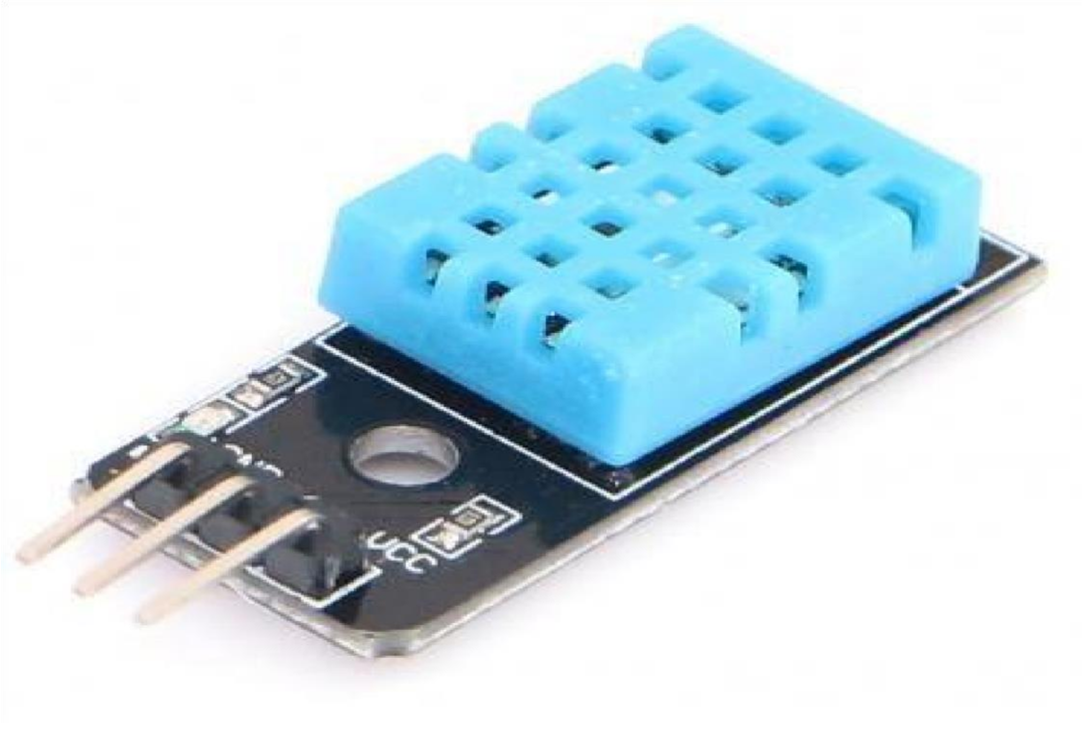
Operating specifications:

(1) Power and Pins Power's voltage should be 3-5.5V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status.One capacitor valued 100nF can be added between VDD and GND for power filtering.

(2) Communication and signal Single-bus data is used for communication between MCU and DHT11.



DHT11
Hunidity and temperature sensor

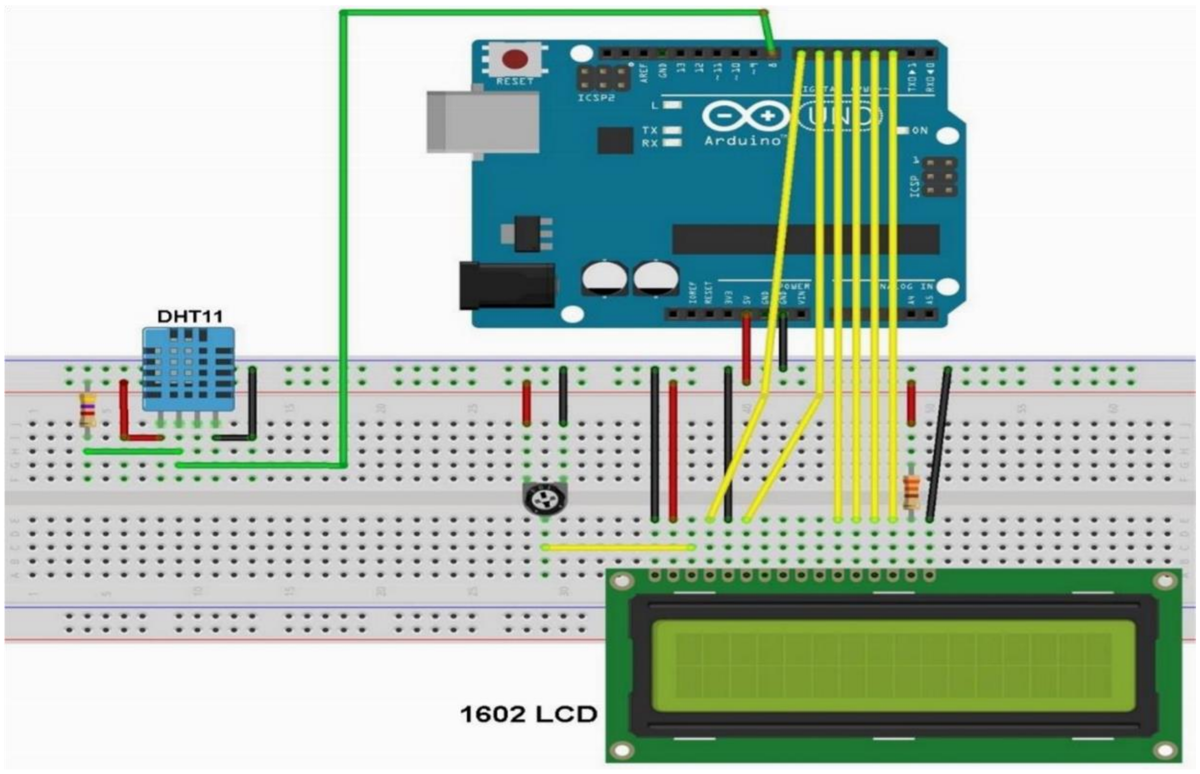


Sample Code

```
#define DHT11_PIN 0      // define anlog  port 0

byte read_dht11_dat()
{ byte i = 0; byte
result=0;   for(i=0;
i< 8; i++)
{
while(!(PINC & _BV(DHT11_PIN)))
{};  // wait  forever until anlog input port 0 is '1'   (NOTICE: PINC reads all the analog input ports
//and  _BV(X) is the macro operation which pull up positon 'X'to '1' and the rest positions to '0'. it is equivalent to
1<    delayMicroseconds(30);
if(PINC & _BV(DHT11_PIN))  //if analog input port 0 is still '1' after 30 us
result |=(1<<(7-i));     //this position is 1 while((PINC &
_BV(DHT11_PIN)));  // wait '1' finish
}
return result;
}



void setup()
{
DDRC |= _BV(DHT11_PIN);   //let analog port 0 be output port
PORTC |= _BV(DHT11_PIN);  //let the initial value of this port be '1'
Serial.begin(9600);
Serial.println("Ready");
}

void loop()
{ byte dht11_dat[5];
byte dht11_in; byte i;//
start condition

PORTC &= ~_BV(DHT11_PIN);    // 1. pull-down i/o pin for 18ms delay(18);
PORTC |= _BV(DHT11_PIN);     // 2. pull-up i/o pin for 40us delayMicroseconds(1);
DDRC &= ~_BV(DHT11_PIN);     //let analog port 0 be input port delayMicroseconds(40);

dht11_in = PINC & _BV(DHT11_PIN);  // read only the input port 0
if(dht11_in) {
Serial.println("dht11 start condition 1 not met"); // wait for DHT response signal: LOW
delay(1000); return; }
delayMicroseconds(80); dht11_in = PINC
& _BV(DHT11_PIN); //
if(!dht11_in) {
Serial.println("dht11 start condition 2 not met");  //wair for second response signal:HIGH return;
delayMicroseconds(80);// now ready for data reception for (i=0; i<5; i++)
{  dht11_dat[i] = read_dht11_dat();}  //recieved 40 bits data. Details are described in datasheet

DDRC |= _BV(DHT11_PIN);       //let analog port 0 be output port after all the data have been received
PORTC |= _BV(DHT11_PIN);      //let the  value of this port be '1' after all the data have been received
byte dht11_check_sum = dht11_dat[0]+dht11_dat[1]+dht11_dat[2]+dht11_dat[3];// check check_sum
if(dht11_dat[4]!= dht11_check_sum)
{
Serial.println("DHT11 checksum error");
}
Serial.print("Current humdity = ");
Serial.print(dht11_dat[0], DEC);
Serial.print(".");
Serial.print(dht11_dat[1], DEC);
Serial.print("%  ");
Serial.print("temperature = ");
Serial.print(dht11_dat[2], DEC);
Serial.print(".");
Serial.print(dht11_dat[3], DEC);
Serial.println("C  ");
delay(2000);  //fresh time
}
```

We use this sensor at project to calculate the temp. & hum

With Lcd (16*2) and potintialmetter to show deggre In the place.



(2-2) motion sensor(PIR):

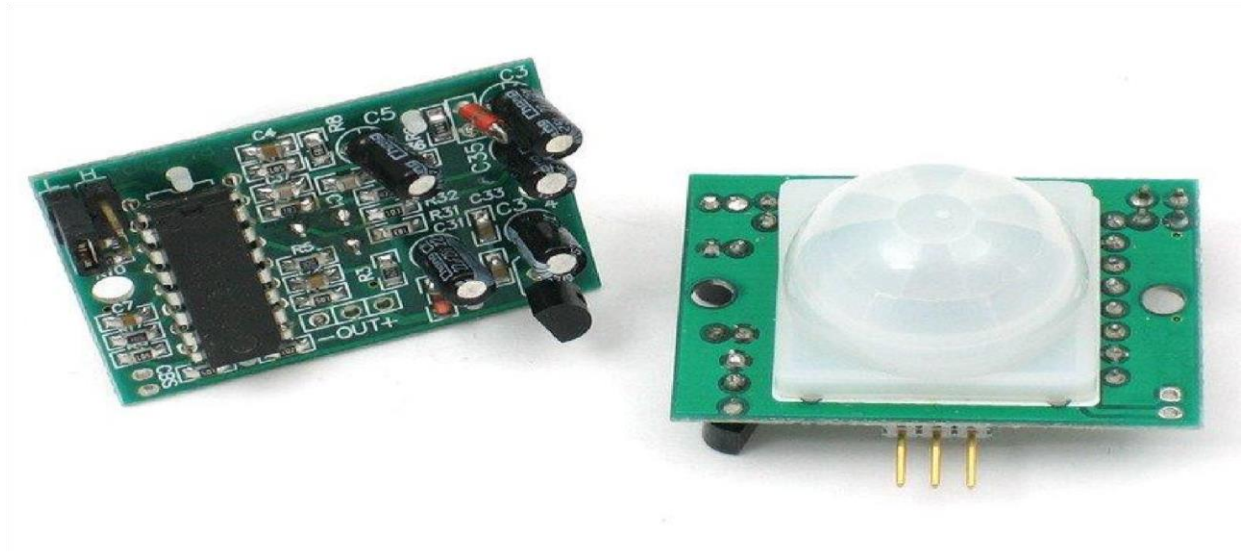
Overview:

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

Connecting PIR sensors to a microcontroller is really simple. The PIR acts as a digital output, it can be high voltage or low voltage, so all you need to do is listen for the pin to flip high (detected) or low (not detected) by listening on a digital input on your Arduino

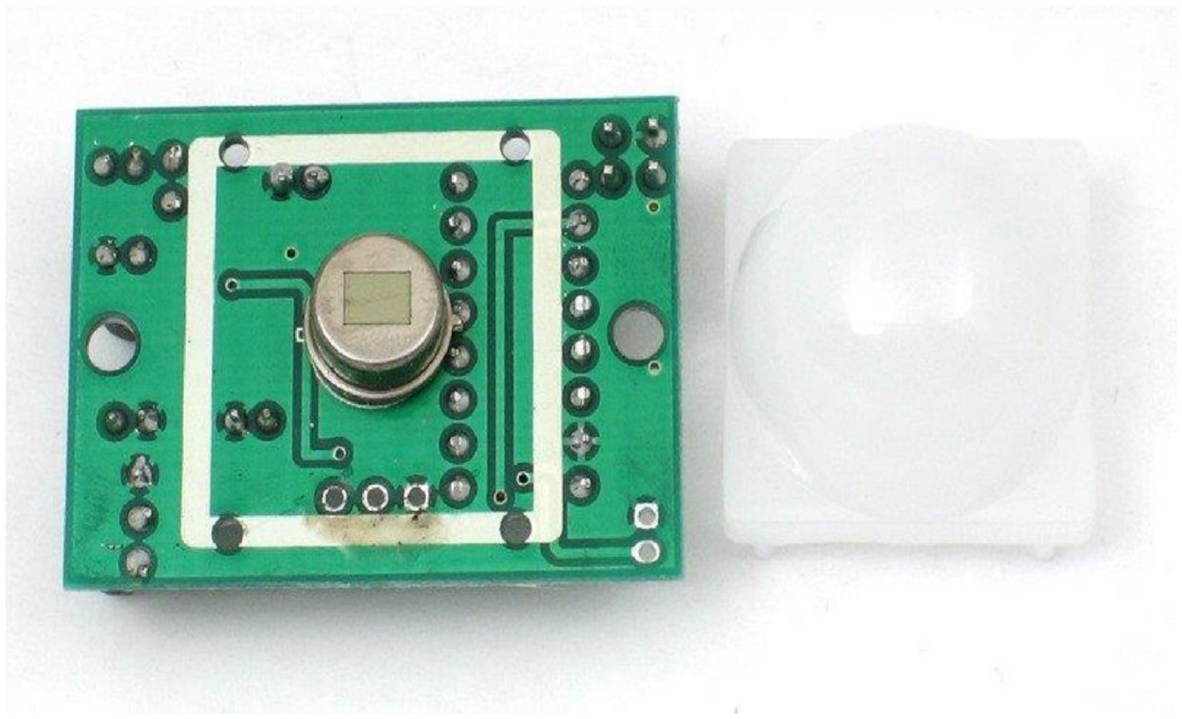
Its likely that you'll want reriggering, so be sure to put the jumper in the H position!

Power the PIR with 5V and connect ground to ground. Then connect the output to a digital pin. In this example we'll use pin 2.



PIRs are basically made of a pyroelectric sensor (which you can see below as the round metal can with a rectangular crystal in the center), which can detect levels of infrared radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is actually split in two halves.

The reason for that is that we are looking to detect motion (change) not average IR levels. The two halves are wired up so that they cancel each other out. If one half sees more or less IR radiation than the other, the output will swing high or low.

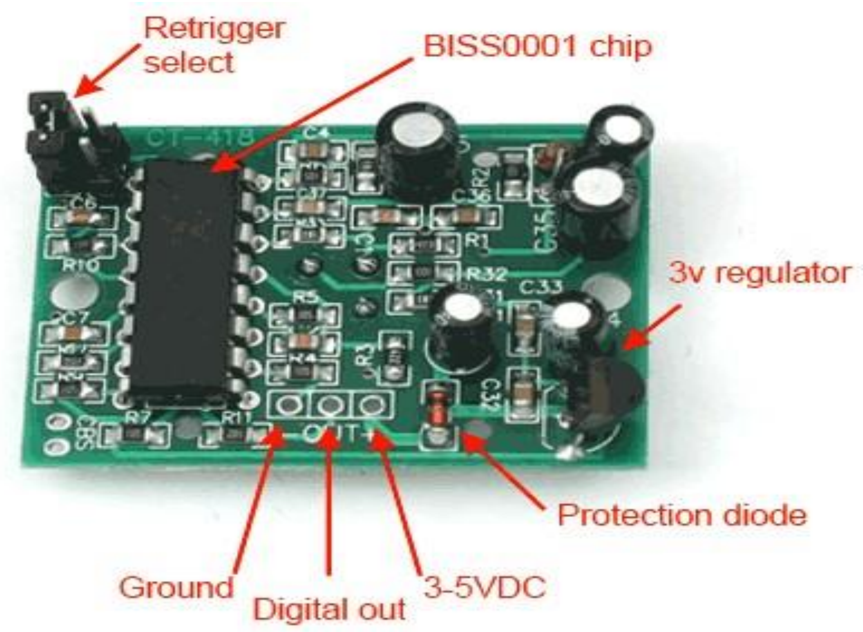


Along with the pyroelectic sensor is a bunch of supporting circuitry, resistors and capacitors.

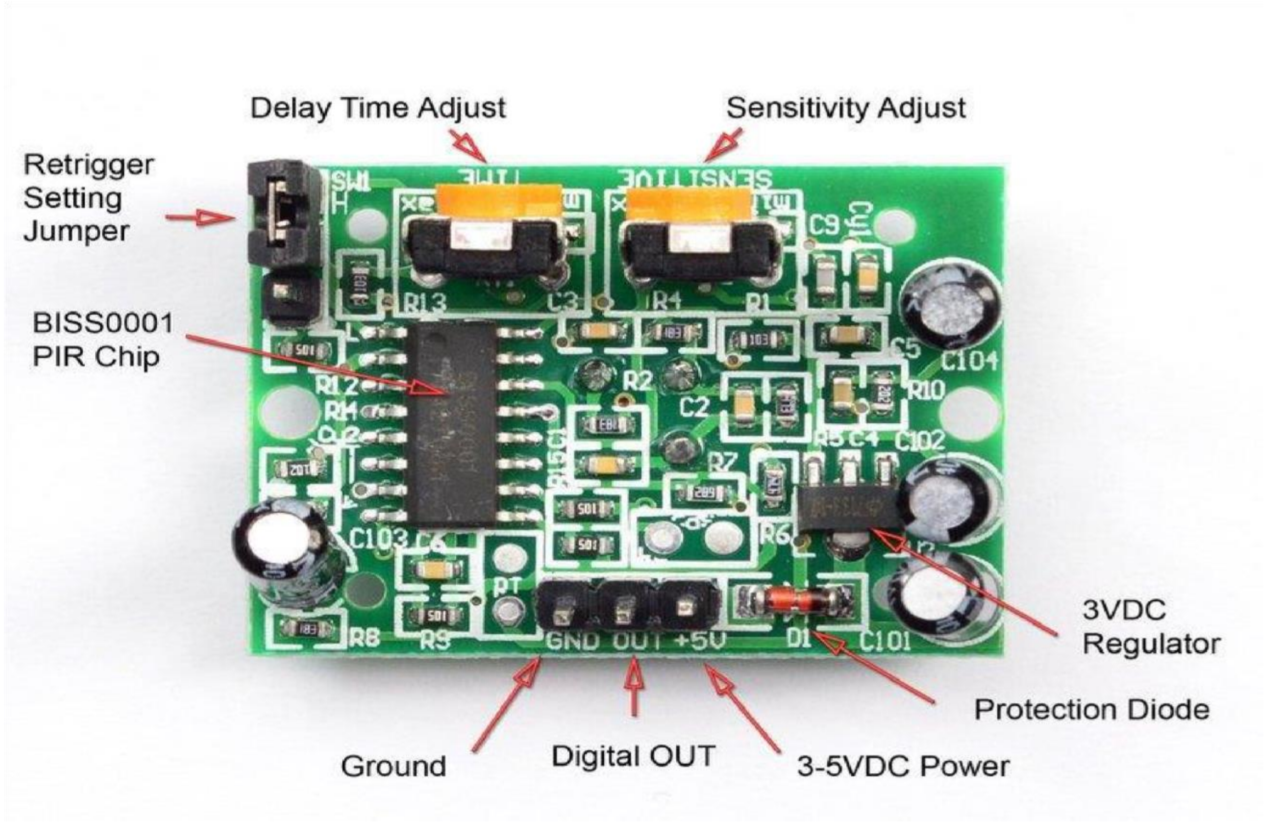
It seems that most small hobbyist sensors use the BISS0001 ("Micro Power PIR Motion Detector IC"), undoubtedly a very inexpensive chip.

This chip takes the output of the sensor and does some minor processing on it to emit a digital output pulse from the analog sensor

Our older PIRs looked like this:



Our new PIRs have more adjustable settings and have a header installed in the 3-pin ground/out/power pads



What does a PIR sensor detect?

As we all know that PIR sensors can be also refer to PID, which is short for passive infrared detectors. As you have learned about the technical term in the first part, PIR sensor can detect infrared radiation which is emitted by particles.

Generally, PIR can detect animal/human movement in a requirement range, which is determined by the spec of the specific sensor. The detector itself does not emit any energy but passively receives it, detects infrared radiation from the environment. Once there is infrared radiation from the human body/ particle with temperature, focusing on the optical system causes the pyroelectric device to generate a sudden electrical signal and an alarm is issued.

How does PIRs work?

The passive infrared alarm does not radiate energy to space but relies on receiving infrared radiation from the human body to make an alarm. Any object with temperature is constantly radiating infrared rays to the outside world. The surface temperature of the human body is 36-27 ° C, and most of its radiant energy is concentrated in the wavelength range of 8-12 um.

Passive infrared alarms can be classified into infrared detectors (infrared probes) and alarm control sections. The most widely used infrared detector is a pyroelectric detector, which is used as a sensor for converting human infrared radiation into electricity. If the human infrared radiation is directly irradiated on the detector, it will, of course, cause a temperature change to output a signal, but in doing so, the detection distance will not be far. In order to lengthen the detection distance of the detector, an optical system must be added to collect the infrared radiation, usually using a plastic optical reflection system or a Fresnel lens made of plastic as a focusing system for infrared radiation.

In the detection area, the infrared radiation energy of the human body through the clothing is received by the lens of the detector and focused on the pyroelectric sensor. When the human body (intruder) moves in this surveillance mode, it enters a certain field of view in sequence and then walks out of the field of view. The pyroelectric sensor sees the moving human body for a while and then does not see it, so the human body The infrared radiation constantly changes the temperature of the pyroelectric material so that it outputs a corresponding signal, which is the alarm signal.

The Range of PIR Sensor?

Indoor passive infrared: Detection distances range from 25 cm to 20 m.

Indoor curtain type: The detection distance ranges from 25 cm to 20 m.

Outdoor passive infrared: The detection distance ranges from 10 meters to 150 meters.

Outdoor passive infrared curtain detector: distance from 10 meters to 150 meters

Grove – PIR Motion Sensor

This Grove – PIR Motion Sensor(Passive Infrared Sensor) can detect infrared signal caused by motion. If the PIR sensor notices the infrared energy, the motion detector is triggered and the sensor outputs HIGH on its SIG pin. The detecting range and response speed can be adjusted by 2 potentiometers soldered on its circuit board, The response speed is from 0.3s – 25s, and max 6 meters of detecting range.

This is easy to use a motion sensor with Grove compatible interface. By Simply connecting it to Base Shield and programming it, it can be used as a suitable motion detector for Arduino project

Sample code :

```
/*
* PIR sensor tester
*/

int ledPin = 13; // choose the pin for the LED
int inputPin = 2; // choose the input pin (for PIR sensor)
int pirState = LOW; // we start, assuming no motion detected
int val = 0; // variable for reading the pin status

void setup() {
pinMode(ledPin, OUTPUT); // declare LED as output
pinMode(inputPin, INPUT); // declare sensor as input

Serial.begin(9600);
}

void loop(){
val = digitalRead(inputPin); // read input value
if (val == HIGH) { // check if the input is HIGH
digitalWrite(ledPin, HIGH); // turn LED ON
if (pirState == LOW) {
// we have just turned on
Serial.println("Motion detected!");
// We only want to print on the output change, not state
pirState = HIGH;
}
} else {
digitalWrite(ledPin, LOW); // turn LED OFF
if (pirState == HIGH){
// we have just turned of
Serial.println("Motion ended!");
// We only want to print on the output change, not state
pirState = LOW;
} } }
```

And we used this sensor to Calculate the number of objects passing through the doors and appear on the 7-segment to counter

(3-2) LDR SENSOR & LASER:”laser system”

To make this system we need :

Hardware:

- LED
- Laser
- Photoresistor
- Piezo
- Buzzer
- Breadboards
- Arduino mega
- Resistor 10Ω
- Resistor220Ω
- Resistor for led.
- jumper wire.

Software:

- Arduino IDE

What is a Light Dependent Resistor?

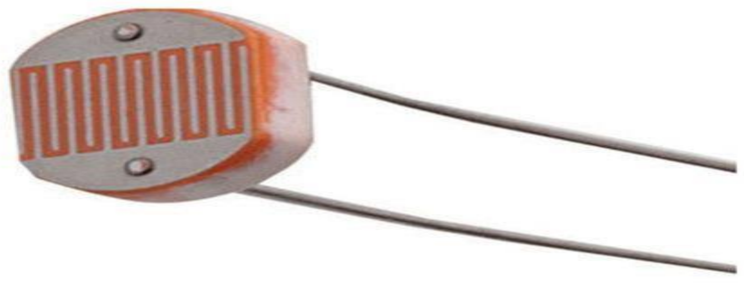
An LDR , photocell, photoconductor. It is a one type of resistor whose resistance varies depending on the amount of light falling on its surface. When the light falls on the resistor, then the resistance changes.

These resistors are often used in many circuits where it is required to sense the presence of light.

These resistors have a variety of functions and resistance. For instance, when the LDR is in darkness, then it can be used to turn ON a light or to turn OFF a light when it is in the light.

A typical light dependent resistor has a resistance in the darkness of 1MOhm, and in the brightness a resistance of a couple of KOhm.

Majority of street lights, outdoor lights, and a number of indoor home appliances are typically operated and maintained manually in many occasions. This is not only risky, however additionally it leads to wastage of power with the negligence of personnel or uncommon circumstances in controlling these electrical appliances ON and OFF. Hence, we can utilize the light sensor circuit for automatic switch OFF the loads based on daylight’s intensity by employing a light sensor. This article discusses in brief about what is a light dependent resistor, how to make a light dependent resistor circuit and its applications.



Working Principle of LDR:

This resistor works on the principle of photo conductivity. It is nothing but, when the light falls on its surface, then the material conductivity reduces and also the electrons in the valence band of the device are excited to the

conduction band. These photons in the incident light must have energy greater than the band gap of the semiconductor material.This makes the electrons to jump from the valence band to conduction

These devices depend on the light, when light falls on the LDR then the resistance decreases, and increases in the dark.When a LDR is kept in the dark place, its resistance is high and, when the LDR is kept in the light its resistance will decrease

Security System Controlled by An Electronic Eye

This security system controlled by an electronic eye project is based on photo sensing arrangement. The proposed system uses a 14-stage ripple carry binary counter to sense the intensity of light using LDR. The o/p makes a relay and buzzer for the required action. This project is very useful to deter burglars from shopping malls, banks and jewelry shops, etc.

This project uses a light dependent resistor. When light falls on the LDR sensor, then the resistance of the sensor decreases, which lead to activate an alarm to give an alert to the user. This project is suitable in the application of providing security system for lockers, cash boxes which can be found in the banks, shopping malls, jewel shops.

The circuit of this project is placed inside of the cash box in shopping malls or inside of the lockers in banks in such a way that, when a burglar opens the cash box or locker and uses a torch light to search the valuables. When the light falls on the circuit which includes an electronic eye and gives a command to the ripple counter. This triggers the alarm and shows a burglary attempt. A lamp is also used to indicate the theft when light falls on the sensor.

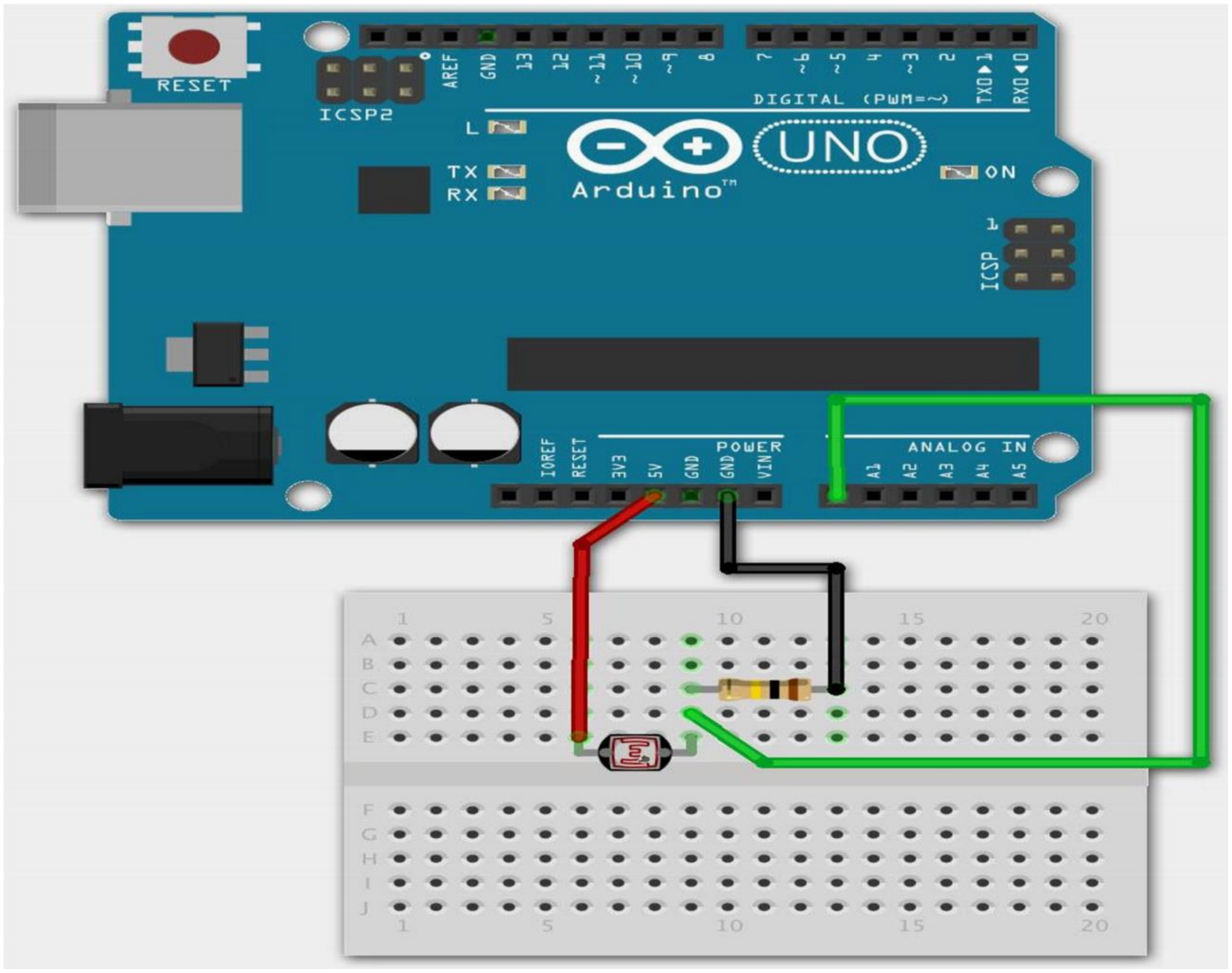
In future, this project can be developed by using a GSM modem and also a microcontroller. This modem can be interfaced to send an SMS to the user in case of burglary

LDR on an Arduino Analog Pin :

This would be the most used and most obvious way to use an LDR, as it fluctuates resistance, producing many values.

In this setup we will make it that the value read from the Analog pin will actually increase as light increases. For this we use a tiny circuit that pushes power through the LDR. Since the LDR decreases its resistance as light increases, more “power” will pass through it to the Analog pin, which results in the Arduino “reading” a higher value.

The analog pin will read values between 0 and 1023, so it converts the analog signal to a digital representation – also called Analog Digital Converter (AD or ADC), which is build into the Arduino.



Source code:

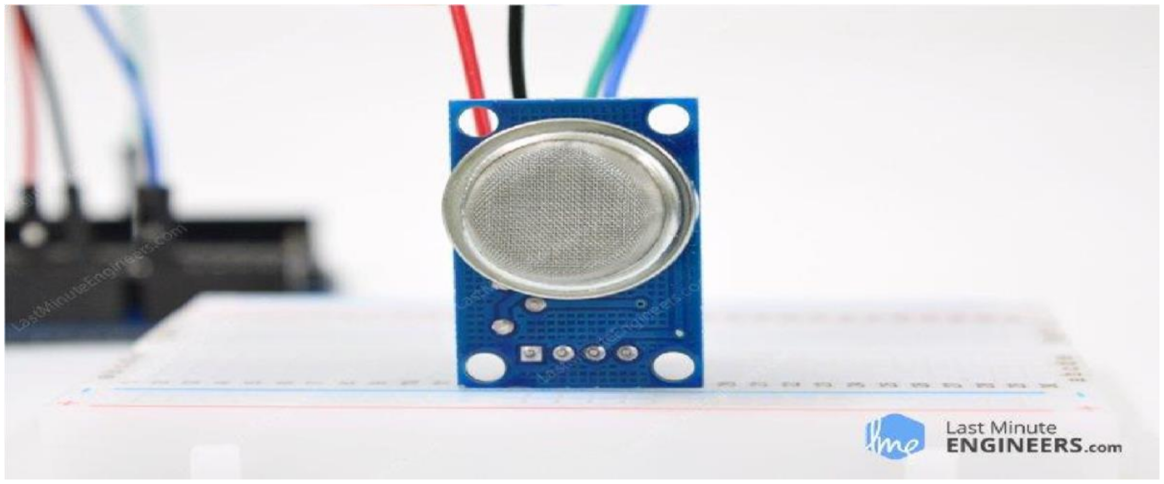
```
#define LDRpin A0 // pin where we connected the LDR and the resistor

int LDRValue = 0;     // result of reading the analog pin

void setup() {
  Serial.begin(9600); // sets serial port for communication
}

void loop() {
  LDRValue = analogRead(LDRpin); // read the value from the LDR
Serial.println(LDRValue);         // print the value to the serial port
delay(100);                    // wait a little }

}
```

We use it with “buzzer” to identify foreign objects and alert them as they pass

(4-2) gas and smoke sensor (MQ2):

Give your next Arduino project a nose for gasses with the MQ2 Gas Sensor Module. This is a robust Gas sensor suitable for sensing LPG, Smoke, Alcohol, Propane, Hydrogen, Methane and Carbon Monoxide concentrations in the air.

If you are planning on creating an indoor air quality monitoring system; breath checker or early fire detection system, MQ2 Gas Sensor Module is a great choice.



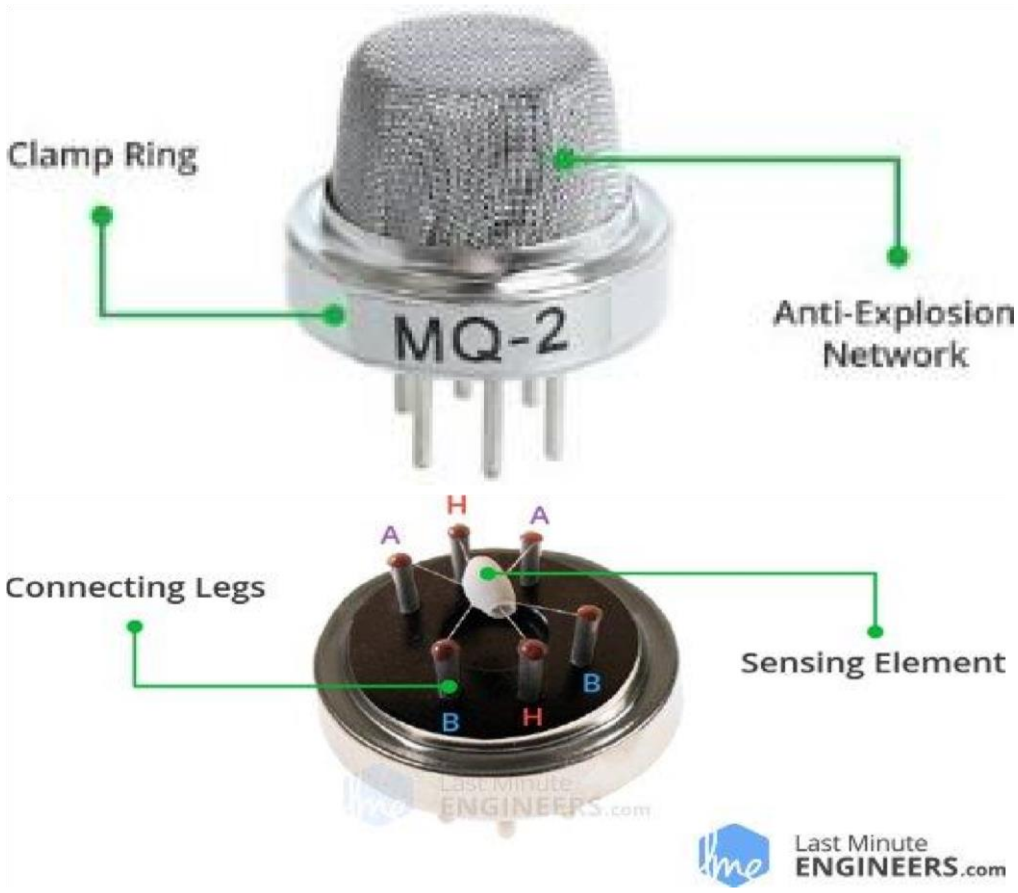
What is MQ2 Gas Sensor?

MQ2 is one of the commonly used gas sensors in MQ sensor series. It is a Metal Oxide Semiconductor (MOS) type Gas Sensor also known as Chemiresistors as the detection is based upon change of resistance of the sensing material when the Gas comes in contact with the material. Using a simple voltage divider network, concentrations of gas can be detected.

Internal structure of MQ2 Gas Sensor

The sensor is actually enclosed in two layers of fine stainless steel mesh called Anti-explosion network. It ensures that heater element inside the sensor will not cause an explosion, as we are sensing flammable gases.

It also provides protection for the sensor and filters out suspended particles so that only gaseous elements are able to pass inside the chamber. The mesh is. bound to rest of the body via a copper plated clamping ring



How does a gas sensor work?

When tin dioxide (semiconductor particles) is heated in air at high temperature, oxygen is adsorbed on the surface. In clean air, donor electrons in tin dioxide are attracted toward oxygen which is adsorbed on the surface of the sensing material. This prevents electric current flow.

In the presence of reducing gases, the surface density of adsorbed oxygen decreases as it reacts with the reducing gases. Electrons are then released into the tin dioxide, allowing current to flow freely through the sensor.

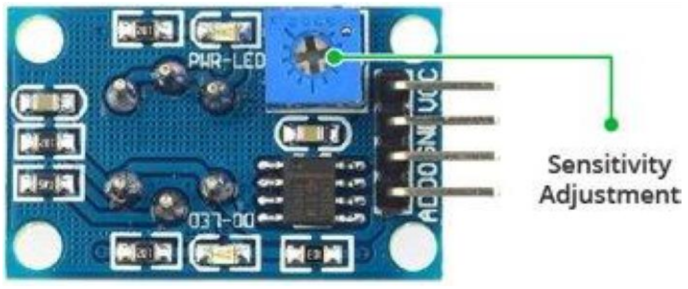
Hardware Overview – MQ2 Gas Sensor Module

Since MQ2 Gas Sensor is not breadboard compatible, we do recommend this handy little breakout board. It's very easy to use and comes with two different outputs. It not only provides a binary indication of the presence of combustible gases but also an analog representation of their concentration in air.

The analog output voltage provided by the sensor changes in proportional to the concentration of smoke/gas. The greater the gas concentration, the higher is the output voltage; while lesser gas concentration results in low output voltage. The following animation illustrates the relationship between gas concentration and output voltage. The analog signal from MQ2 Gas sensor is further fed to LM393 High Precision Comparator (soldered on the bottom of the module), of course to digitize the signal. Along with the comparator is a little potentiometer you can turn to adjust the sensitivity of the sensor. You can use it to adjust the concentration of gas at which the sensor detects it.

Calibrate MQ2 Gas Sensor Module

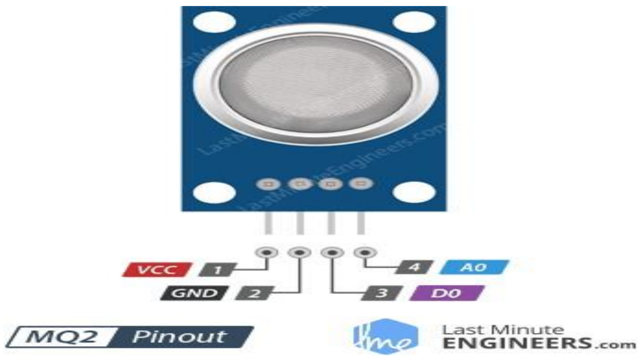
To calibrate the gas sensor you can hold the gas sensor near smoke/gas you want to detect and keep turning the potentiometer until the Red LED on the module starts glowing. Turn the screw clockwise to increase sensitivity or anticlockwise to decrease sensitivity.



The comparator on the module continuously checks if the analog pin (A0) has hit the threshold value set by potentiometer. When it crosses the threshold, the digital pin (D0) will go HIGH and signal LED turns on. This setup is very useful when you need to trigger an action when certain threshold is reached. For example, when the smoke crosses a threshold, you can turn on or off a relay or instruct your robot to blow air/sprinkle water. You got the idea!

MQ2 Gas Sensor Module Pinout

Now let's have a look at the pinout.



VCC supplies power for the module. You can connect it to 5V output from your Arduino.

GND is the Ground Pin and needs to be connected to GND pin on the Arduino.
D0 provides a digital representation of the presence of combustible gases.
provides analog output voltage in proportional to the concentration of smoke/gas.

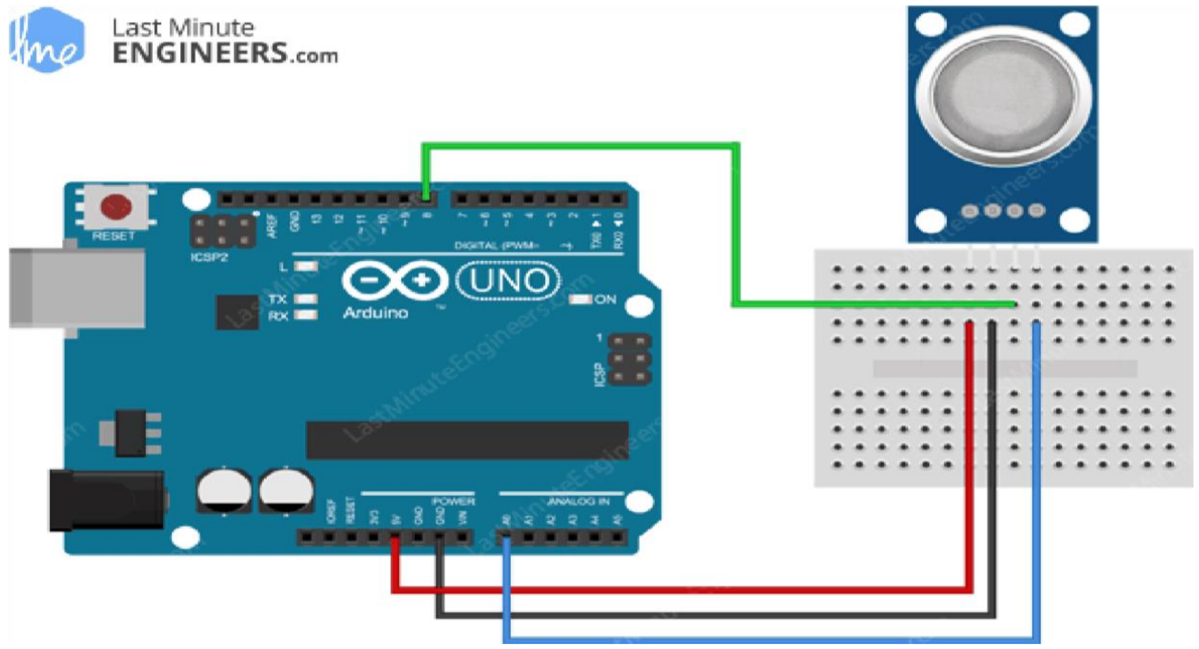
Wiring – Connecting MQ2 Gas Sensor Module to Arduino UNO

Now that we have a complete understanding of how MQ2 Gas sensor works, we can begin hooking it up to our Arduino!

Connecting the MQ2 Gas sensor module to the Arduino is pretty easy. Start by placing the sensor on to your breadboard. Connect VCC pin to the 5V pin on the Arduino and connect GND pin to the Ground pin on the Arduino.

Connect D0 output pin on the module to Digital pin#8 on the Arduino and A0 output pin on the module to Analog pin#0 on the Arduino.

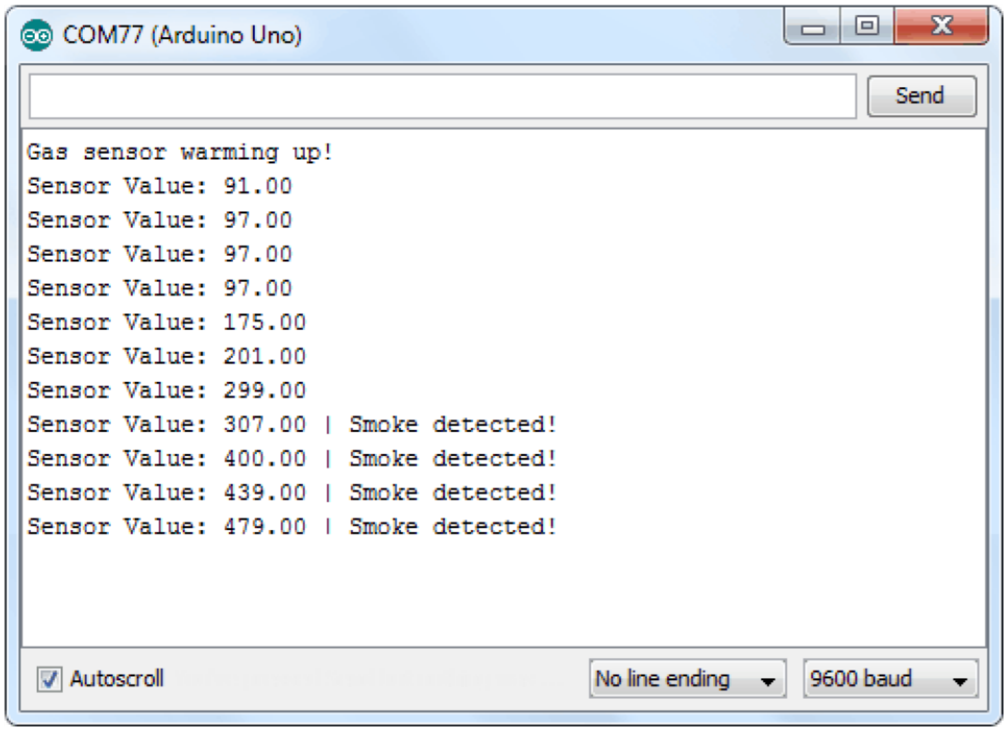
When you're done you should have something that looks similar to the illustration shown below.



Source code:

```
#define MQ2pin (0)
 float sensorValue;  //variable to store sensor value

 void setup()

{
  Serial.begin(9600); // sets the serial port to 9600
Serial.println("Gas sensor warming up!");   delay(20000);
// allow the MQ-6 to warm up

}  void loop() {   sensorValue = analogRead(MQ2pin); // read
analog input pin 0


  Serial.print("Sensor Value: ");
  Serial.print(sensorValue);
     if(sensorValue >
300)
  {
    Serial.print(" | Smoke detected!");
  }

  Serial.println("");   delay(2000); //
wait 2s for next reading

}
```

The output on the serial monitor looks like:



```
Gas sensor warming up!
Sensor Value: 91.00
Sensor Value: 97.00
Sensor Value: 97.00
Sensor Value: 97.00
Sensor Value: 175.00
Sensor Value: 201.00
Sensor Value: 299.00
Sensor Value: 307.00 | Smoke detected!
Sensor Value: 400.00 | Smoke detected!
Sensor Value: 439.00 | Smoke detected!
Sensor Value: 479.00 | Smoke detected!
```

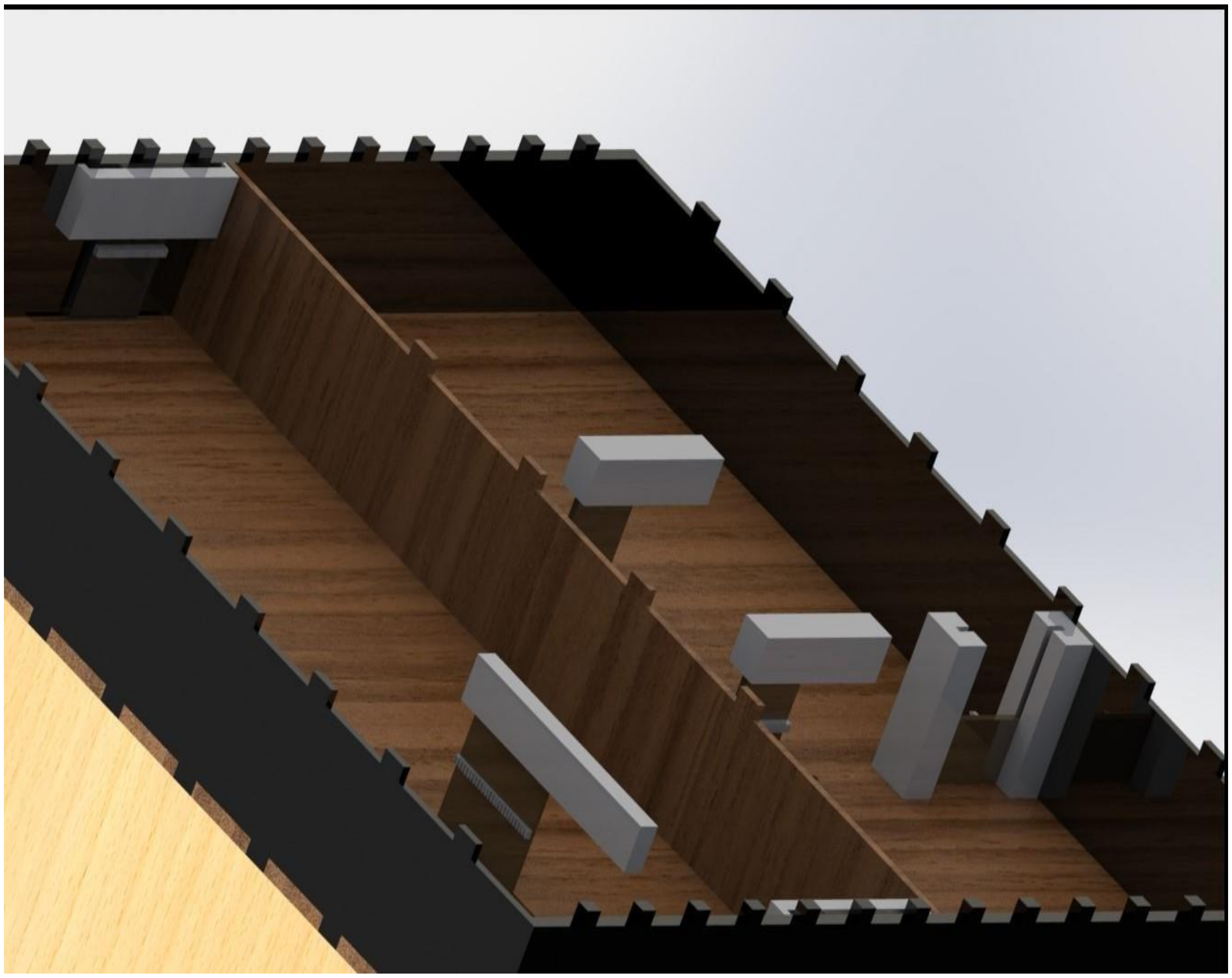
.

And we use it to control the smoke and gas in place and if the find smoke or high at gas level make alarm to me .

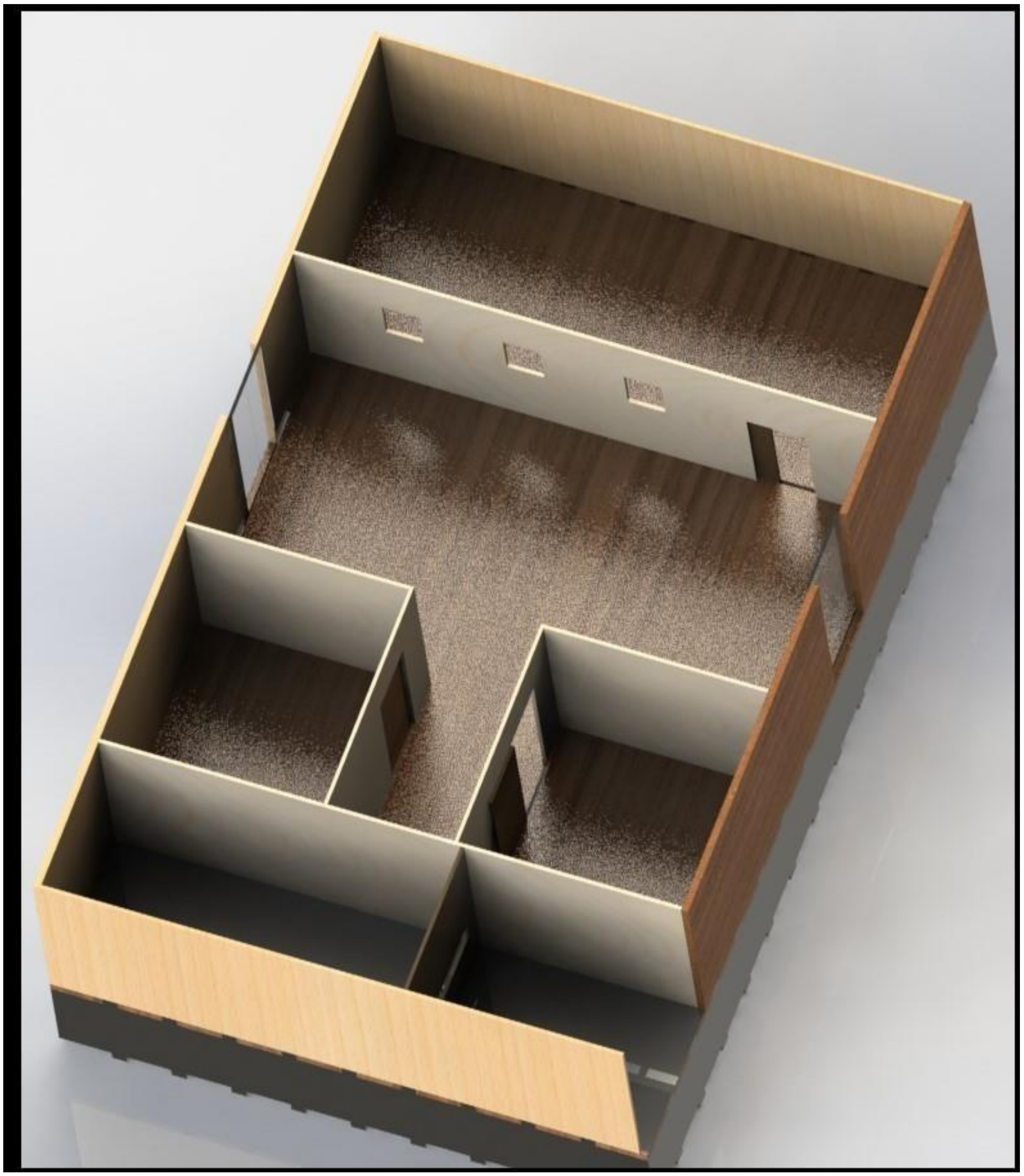
Chapter(3)

Maquette Design

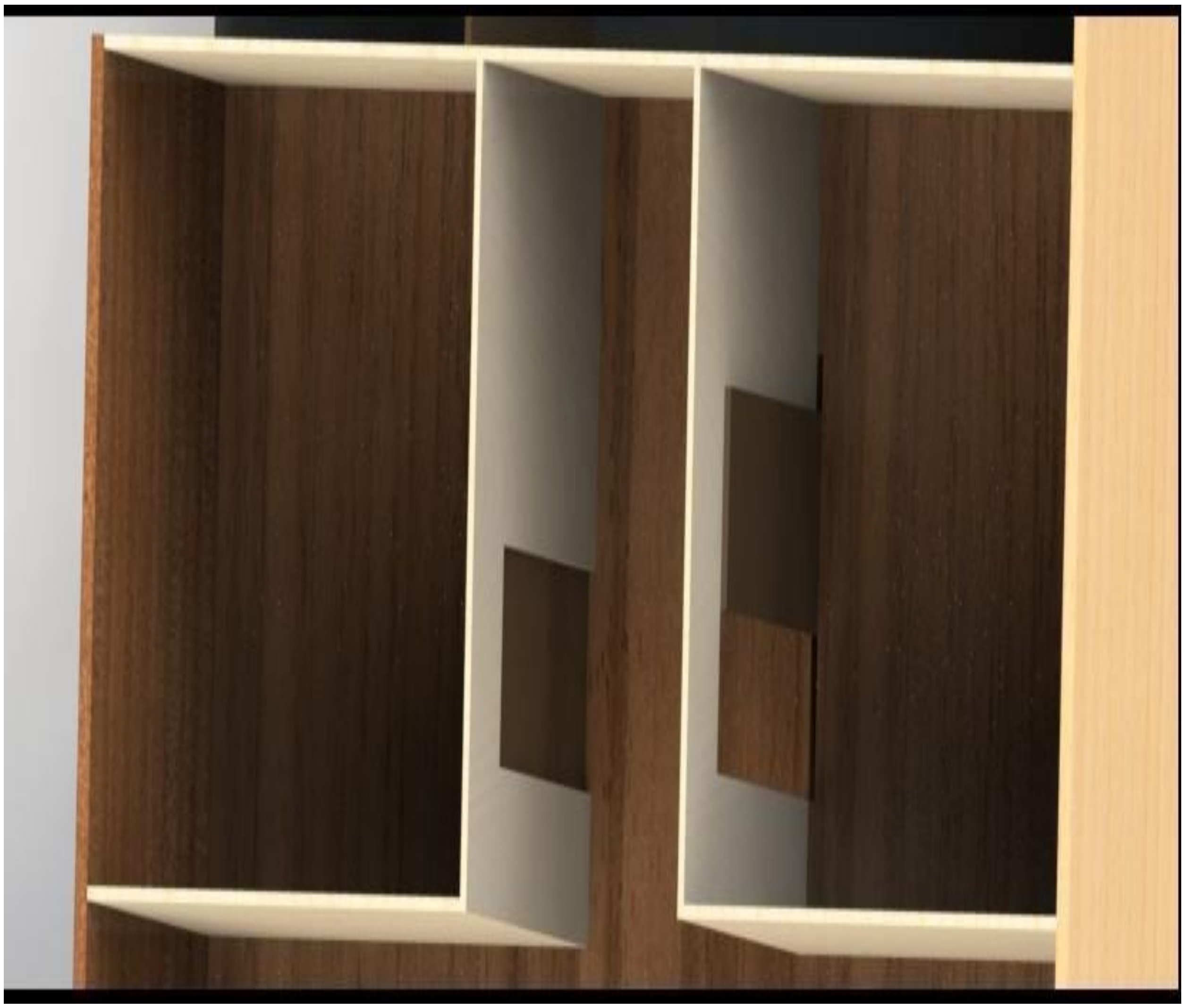
(1-3)Maquette design:



Under shape



Top shape



Side rooms

Main doors

Employees room

Garage to parking

(3-2)circuit code:

```
#include <dht.h>   // includes temperature/humadity Sensor Library

#include <LiquidCrystal.h> // includes the LiquidCrystal Library

#define DHT_Pin 2  // set digital pin 2 as input pin of DHT

#define LDR_Pin A0  // set analog pin A0 as input pin of LDR

#define MQ2_Pin A5  //set analog pin A5 as input pin of MQ2

#define PIR_pin 8  //set digital pin 2 as input pin of PIR

#define Buzzer_Pin 13 //set digital pin 13 as output pin of buzzer

#define yellowLed 12 //set digital pin 12 as output pin of Yellow Led

#define RedLed 11 //set digital pin 11 as output pin of Red Led

#define PIRLed 14 //set digital pin 13 as output pin of PIR motion led

#define buzzerPin 10  //set digital pin 10 as output pin of buzzer

#define check_time 500 const int rs=1, en=3, d4=4, d5=5, d6=6, d7=7; //

Creates an LCD object LiquidCrystal lcd(rs, en, d4, d5, d6, d7); dht

DHT;  //structure for temp sensor int sensorThres = 600;  //threshold

value after bypassing buzzer ring int pirState = 0 ; //intial value of PIR

as LOW mode int val=0; // storage variable for reading the pin status

long time1, time2, time3, time4 ; //times to re-check every sensor void

setup()

{

 lcd.begin(16, 2);  // Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display

  // declare every used pin as input / output

 pinMode(LDR_Pin,INPUT);

pinMode(MQ2_Pin, INPUT);

pinMode(Buzzer_Pin,OUTPUT);

pinMode(yellowLed, OUTPUT);

pinMode(RedLed, OUTPUT);   pinMode(PIRLed,

OUTPUT);      pinMode(PIR_pin, INPUT);

pinMode(buzzerPin, OUTPUT);
```

```
  time1=millis();
```

//DHT function

```
void dht()

{

  int readData = DHT.read11(DHT_Pin);   float t = DHT.temperature; // storage variable of

temperature   float h = DHT.humidity; // storage variable of humidity   lcd.setCursor(0,0); // Sets the

location at which subsequent text written to the LCD will be displayed   lcd.print("Temp.: "); //

Prints string "Temp." on the LCD   lcd.print(t); // Prints the temperature value from the sensor

lcd.print(" C"); // Prints C that aims to Celisuis   lcd.setCursor(0,1); // Cursor of the lcd to start

writing from   lcd.print("Humi.: "); // Prints string "Humi." on the LCD   lcd.print(h); // Prints the

humidity value from the sensor   lcd.print(" %"); // Prints % that aims to percent

}
```

// LDR function

```
void ldr()
{
  int ldrval=analogRead(LDR_Pin); // storage variable for the sensor output
if(ldrval>=900) // if the laser been blocked the buzzer rings

  {
    buzz(50); //buzzer tone
  }
}
  /* Function of the buzzer to make a specific tone and delay for
specific time

   */
  void buzz(unsigned char time)
  {
    analogWrite(Buzzer_Pin,170);
delay(time);
analogWrite(Buzzer_Pin,0);
delay(time);

  }
```

// MQ2 function

```
void mq2()
{
  int analogSensor = analogRead(MQ2_Pin);
  if (analogSensor > sensorThres)   // Checks if it has reached the threshold value
 {
   digitalWrite(yellowLed, HIGH);
digitalWrite(RedLed, LOW);

 }
else

 {
   digitalWrite(yellowLed, LOW);
digitalWrite(RedLed, HIGH);

 }
}
```

//PIR function

```
void pir()
{
  val = digitalRead(PIR_pin);  // read input value   if
(val == HIGH) {            // check if the input is HIGH
digitalWrite(PIRLed, HIGH);  // turn LED ON
playTone(300, 160);     delay(50);     if (pirState ==
LOW)

   {
     pirState = HIGH;
   }   }
else

 {
     digitalWrite(PIRLed,  LOW);
playTone(0,  0);           delay(50);
if (pirState == HIGH)

    {
      pirState = LOW;
    }
  }
}
//function to set a time for repeat the motion detection
// duration in mSecs, frequency in hertz void
playTone(long duration, int freq) {     duration *=
1000;     int period = (1.0 / freq) * 1000000;     long
elapsed_time = 0;     while (elapsed_time < duration) {
digitalWrite(buzzerPin,HIGH);
delayMicroseconds(period / 2);
digitalWrite(buzzerPin, LOW);
delayMicroseconds(period / 2);        elapsed_time +=
(period);

    }
}
```