

Intelligent Door Lock System Using IOT

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering Degree in

Electronics and Communication Engineering

by

Kintali Sai Samba Siva 38130107

Kamireddy Praveen Kumar Sai 38130096



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119

MAY-2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

www.sathyabama.ac.in



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Kintali Sai Samba Siva(38130107), Kamireddy Praveen Kumar (38130096) who carried out the project entitled "**Intelligent Door Lock System Using IOT** " under our supervision from November 2021 to May 2022.

Internal Guide

Dr.REKHA CHAKRAVARTHI.,M.E.,Ph.D.,

Head of the Department

Dr. T.RAVI.,M.E.,Ph.D.,

Submitted for Viva voce Examination held on 04/05/2022

Internal Examiner

Dr.S.EMALDA ROSLIN, M.E, Ph.D

External Examiner

Dr.N.R.Krishnamoorthy, M.E, Ph.D

DECLARATION

We **Kintali Sai Samba Siva (RegNo.38130107)** and **Kamireddy Praveen Kumar (RegNo.38130096)** hereby declare that the Project Report entitled " **Intelligent Door Lock System Using IOT**" done by us under the guidance of **Dr.S.Emalda Roslin, M.E, Ph.D** and **Dr. N.R.Krishnamoorthy, M.E, Ph.D** is submitted in complete fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE:

K.Sai sambasiva, k. Praveen kumar

PLACE:

Signature of the candidates

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr. **N.M. NANDHITHA, M.E., Ph.D.**, Dean, School of **ELECTRONICS AND COMMUNICATION ENGINEERING** and Dr. **T. RAVI, M.E., Ph.D.**, Head of the Department, Dept. of **ELECTRONICS AND COMMUNICATION ENGINEERING** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Mrs. **Dr.Rekha Chakravarthi.,M.E.,Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **ELECTRONICS AND COMMUNICATION ENGINEERING** who were helpful in many ways for the completion of the project.

Abstract:

Authentication is one of the significant issues in the era of the information system. Among other things, human face recognition (HFR) is one of the known techniques which can be used for user authentication. As an important branch of biometric verification, HFR has been widely used in many applications, such as video monitoring/surveillance systems, and human-computer interaction. This project proposes a method for automatic door access system using face recognition technique by using embedded C programming and from Haar cascade method. Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones. This is the standalone security device that has been developed by using Nodemcu electronic development board and operated on Battery power supply, wireless internet connectivity by using USB modem. Automatic e-mail notification has been achieved by sending security user's mail to the user e-mail id. This proposed and is more effective, reliable, and this system consumes very less data and power compared to the other existing systems.

CONTENT

CHAPTER 1 :

INTRODUCTION

LITERATURE SURVEY	-	7
PROPOSED SYSTEMS	-	9
BLOCK DIAGRAM	-	9
HARDWARE USED	-	11
VIDEO STREAMING SERVER	-	12

CHAPTER 2 :

GETTING IP ADDRESS	-	16
ACCESSING THE VIDEO STREAMING SERVER	-	17

CHAPTER 3 :

ABOUT IR SENSOR	-	19
MORE ABOUT COMPONENTS	-	21
BOB GEAR MOTOR	-	22

CHAPTER 4 :

AURDINO UNO AND ITS PROGRAMING	-	24
--------------------------------	---	----

CHAPTER 5 :

BLYNK APP	-	39
CREATING NEW PROJECT IN BLYNK	-	41
HARDWARE	-	42
BLYNK APP SETUP	-	44
RUN AND COMPILE	-	49

CHAPTER 6:

CONCLUSION AND RESULT	-	54
-----------------------	---	----

INTRODUCTION :

In the existing system, we are using a password-based door lock system which makes the user memorize passwords the time. So to overcome this drawback we designing a new system with face recognition. In the existing system, the face can be recognized by the USB camera with ESP 32 cam. We finished a face recognition door lock system as we planned. The passcode and face recognition works well. There is high accuracy in recognizing house owner faces and it could realize the matched face image to another ESP 32 Cam in time and give a good output. And it takes a little bit of time to recognize. At all, we are satisfied with the building. It is done in the python programming language. And this project must be inside installed that is Ubuntu. This software should be difficult to operate.

LITERATURE SURVEY

- **There were many types of research done on Door locks Like Pandnya R.Nehete, J.P. Chaudhari, S.R. Pachpande, and Concluded various papers on Future Smart Door Locks Like wifi doors and Lock with GPS, GSM models, OTP Based and Based on many technologies.**

At present, the surveillance of international border areas is a difficult task. the smart door lock automatically detects face recognition and the doors were unlocked by the owner of the house. The information regarding the detection of humans by IR sensor and it capture the image of the person by ESP 32 Cam. in their paper studied the advantages of biometric lock technology over other lock technologies like locks based on RFID. He concluded that biometric lock technology is superior to the other lock technologies.

- **Door Lock system using Cryptographic algorithms based on IOT by Shruti Jalapur, A. Maniyar**

The Doorlock system finds its applications in places such as office buildings, banks, shopping centers, server rooms, labs, and homes. There are also many other applications of a door lock system. In the recent development in technology, the Internet of Things (IoT) has gone through many revolutionary changes in the area of industries, smart home applications, agriculture, health facilities, smartphone and etc. In the area of networking, IoT finds applications related to data confidentiality, control over unauthorized access of confidential data, and also remote access to information. The project aims in designing a Door Lock System based on IoT using cryptographic algorithms. The system designed on the basis of IoT achieves applications of confidentiality, security, and also remote access to data. Sometimes, the data shared over a network might not be secured. Therefore many researchers are taking interest in developing such a system secured from hackers. The cryptographic algorithms which are used in this door lock system will protect the data being communicated from hackers because the data is as important as the valuables behind the door.

Today people are facing more problems about security all over the world, nowadays security is the most essential issue everywhere in the world; so the security of everything gains higher and higher importance in recent years. Here in this paper, trying to reproduce the comprehensive literature study related to the various door locks and gate security systems that are necessary in the fields such as home, industries and vehicle security where possibilities of incursion are increasing day by day. In past days, the research is gone on various door lock security systems like traditional security systems which provide indications using the alarm. Due to the advancement in recent techniques, some door lock security systems are based on microcontrollers, GSM, GPS, many sensors, software like MATLAB, PROTEUS, biometrics like face recognition, Iris scanner, RFID, Smart Card, and password etc. Each system has its own advantages and disadvantages. In most of the systems, the SMS technique is used for communication so the system will become cost-effective, more reliable and it will take less time to deliver

Proposed system:

In this proposed system door access system by using face recognition and along with the blynk alert system has been presented. This system has been used with home door lock access control based on the face recognition method by verifying enrolled facial images.

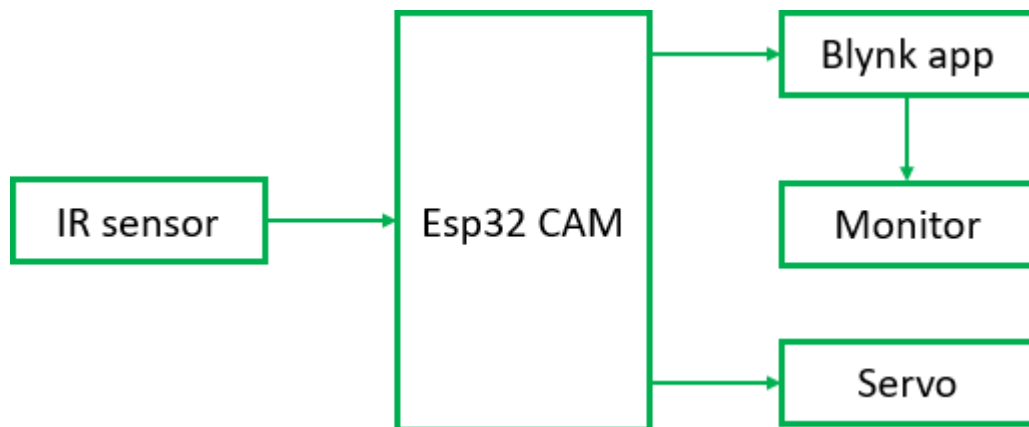
Concern persons will be informed successfully about the person detection via e-mail alert generations along with details attached.

Face recognition is one of the several techniques for recognizing people. Several methods can be used for that purpose. Some of the most common are using PCA or Eigen faces. The Eigen faces are other new techniques simpler to understand the use and implement but also with very good performance.

The Haar Cascades algorithm is one of those algorithms. As we show Haar Cascades has very good performance and is very accurate.

In this system, we can use esp32 cam because it is very accurate and real-time processing unit.

Block diagram:



Hardware used:

Esp32 Cam

IR sensor

Display unit

Boob Gare motor Lock

ESP32-CAM

The ESP32-CAM is a very small camera module with the ESP32-S chip that costs approximately \$10. Besides the OV2640 camera, and several GPIOs to connect peripherals, it also features a microSD card slot that can be useful to store images taken with the camera or to store files to serve to clients.

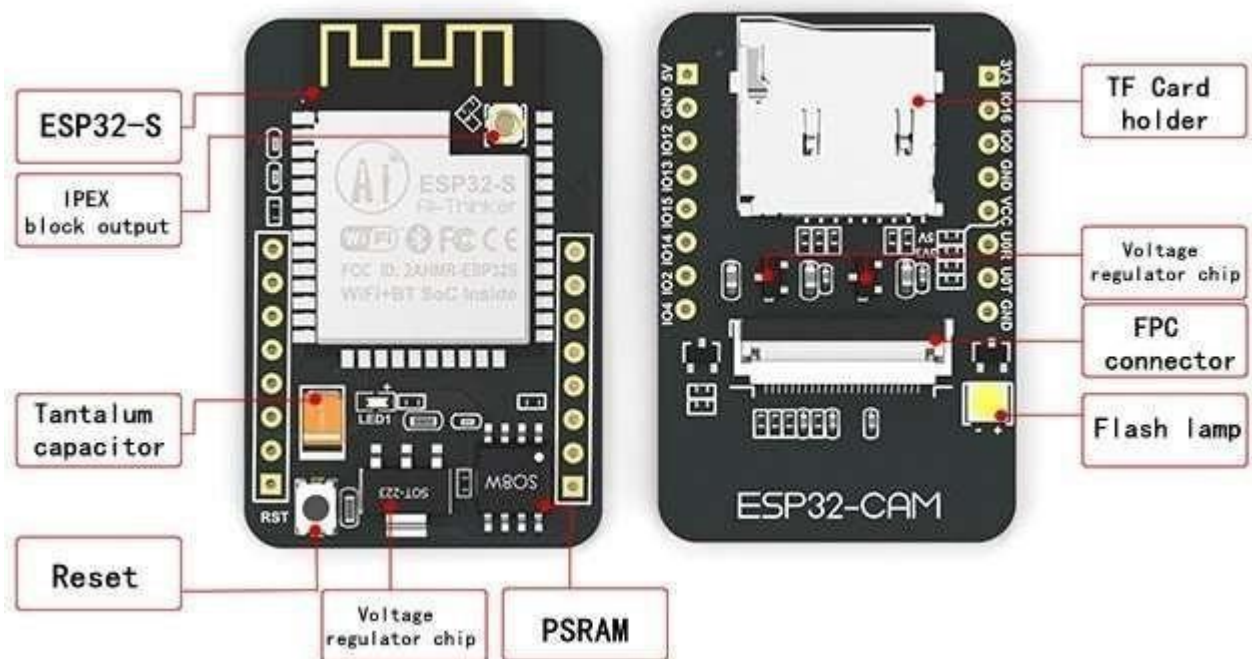
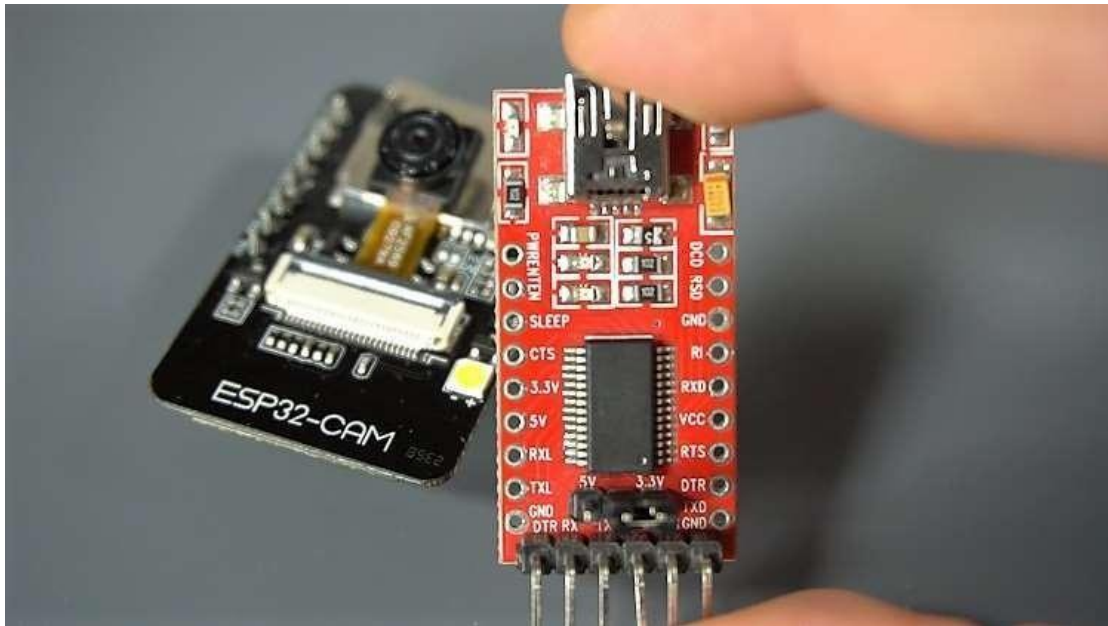


Image source – Seeed Studio

The ESP32-CAM doesn't come with a USB connector, so you need an FTDI programmer to upload code through the U0R and U0T pins (serial pins).



Features

Here is a list of the ESP32-CAM features:

- The smallest 802.11b/g/n Wi-Fi BT SoC module
- Low power 32-bit CPU, can also serve the application processor
- Up to 160MHz clock speed, summary computing power up to 600DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, built-in flash lamp
- Support image WiFi upload
- Support TF card
- Supports multiple sleep modes
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

ESP32-CAM Pinout

The following figure shows the ESP32-CAM pinout (AI-Thinker module).



Image

source – Seeed Studio

There are three **GND** pins and two pins for power: either **3.3V** or **5V**.

GPIO 1 and **GPIO 3** are serial pins. You need these pins to upload code to your board. Additionally, **GPIO 0** also plays an important role, since it determines whether the ESP32 is in flashing mode or not. When **GPIO 0** is connected to **GND**, the ESP32 is in flashing mode.

The following pins are internally connected to the microSD card reader:

- GPIO 14: CLK
- GPIO 15: CMD
- GPIO 2: Data 0
- GPIO 4: Data 1 (also connected to the on-board LED)
- GPIO 12: Data 2
- GPIO 13: Data 3

Video Streaming Server

Follow the next steps to build a video streaming web server with the ESP32-CAM that you can access on your local network.

Important: Make sure you have your Arduino IDE updated as well as the latest version of the ESP32 add-on.

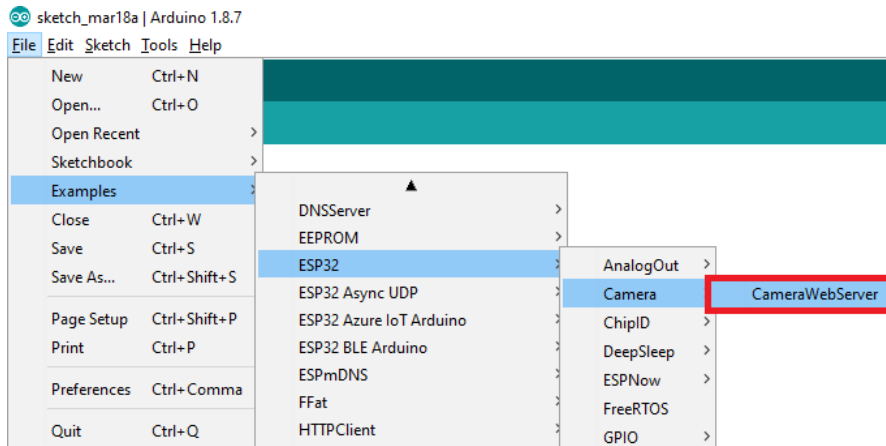
1. Install the ESP32 add-on

In this example, we use Arduino IDE to program the ESP32-CAM board. So, you need to have Arduino IDE installed as well as the ESP32 add-on. Follow one of the next tutorials to install the ESP32 add-on, if you haven't already:

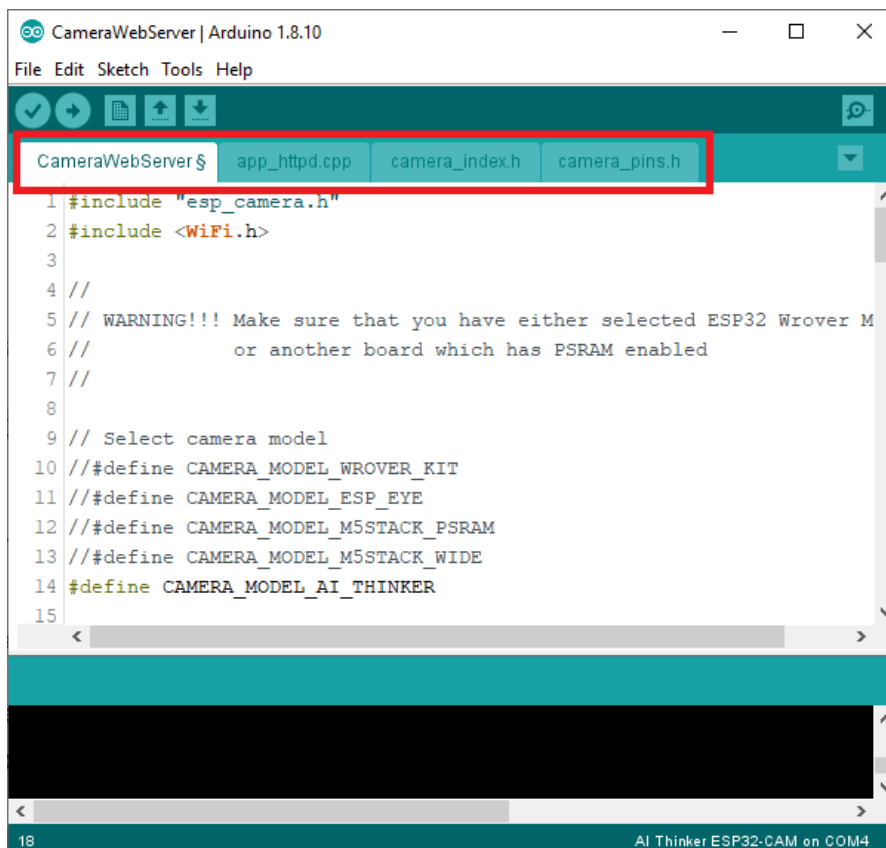
- Installing the ESP32 Board in Arduino IDE (Windows instructions)
- Installing the ESP32 Board in Arduino IDE (Mac and Linux instructions)

2. Camera Webserver Example Code

In your Arduino IDE, go to **File > Examples > ESP32 > Camera** and open the **Camera Webserver** example.



The following code should load.



Before uploading the code, you need to insert your network credentials in the following variables:

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";  
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

Then, make sure you select the right camera module. In this case, we're using the AI-THINKER Model.



So, comment on all the other models and uncomment this one:

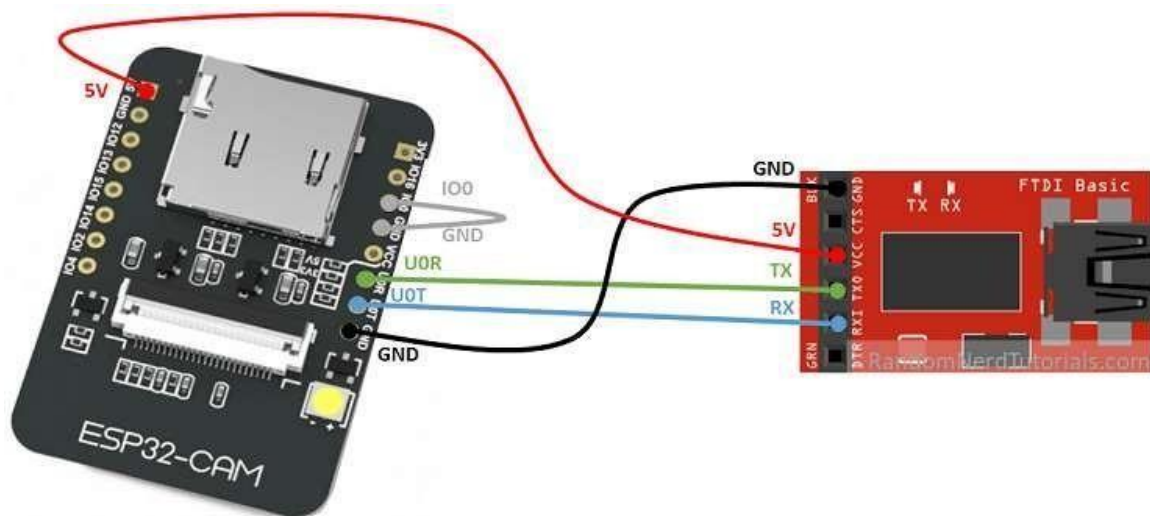
```
// Select camera model  
//#define CAMERA_MODEL_WROVER_KIT  
//#define CAMERA_MODEL_ESP_EYE  
//#define CAMERA_MODEL_M5STACK_PSRAM  
//#define CAMERA_MODEL_M5STACK_WIDE  
#define CAMERA_MODEL_AI_THINKER
```

If none of these correspond to the camera you're using, you need to add the pin assignment for your specific board in the camera_pins.h tab.

Now, the code is ready to be uploaded to your ESP32.

3. ESP32-CAM Upload Code

Connect the ESP32-CAM board to your computer using an FTDI programmer. Follow the next schematic diagram:



Many FTDI programmers have a jumper that allows you to select 3.3V or 5V. Make sure the jumper is in the right place to select 5V.

Important: GPIO 0 needs to be connected to GND so that you're able to upload code.

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

To upload the code, follow the next steps:

- 1) Go to **Tools > Board** and select **AI-Thinker ESP32-CAM**.
- 2) Go to **Tools > Port** and select the COM port the ESP32 is connected to.
- 3) Then, click the upload button to upload the code.



4) When you start to see these dots on the debugging window as shown below, press the ESP32-CAM on-board RST button.

```
esptool.py v2.6-beta1
Serial port COM10
Connecting.....
```

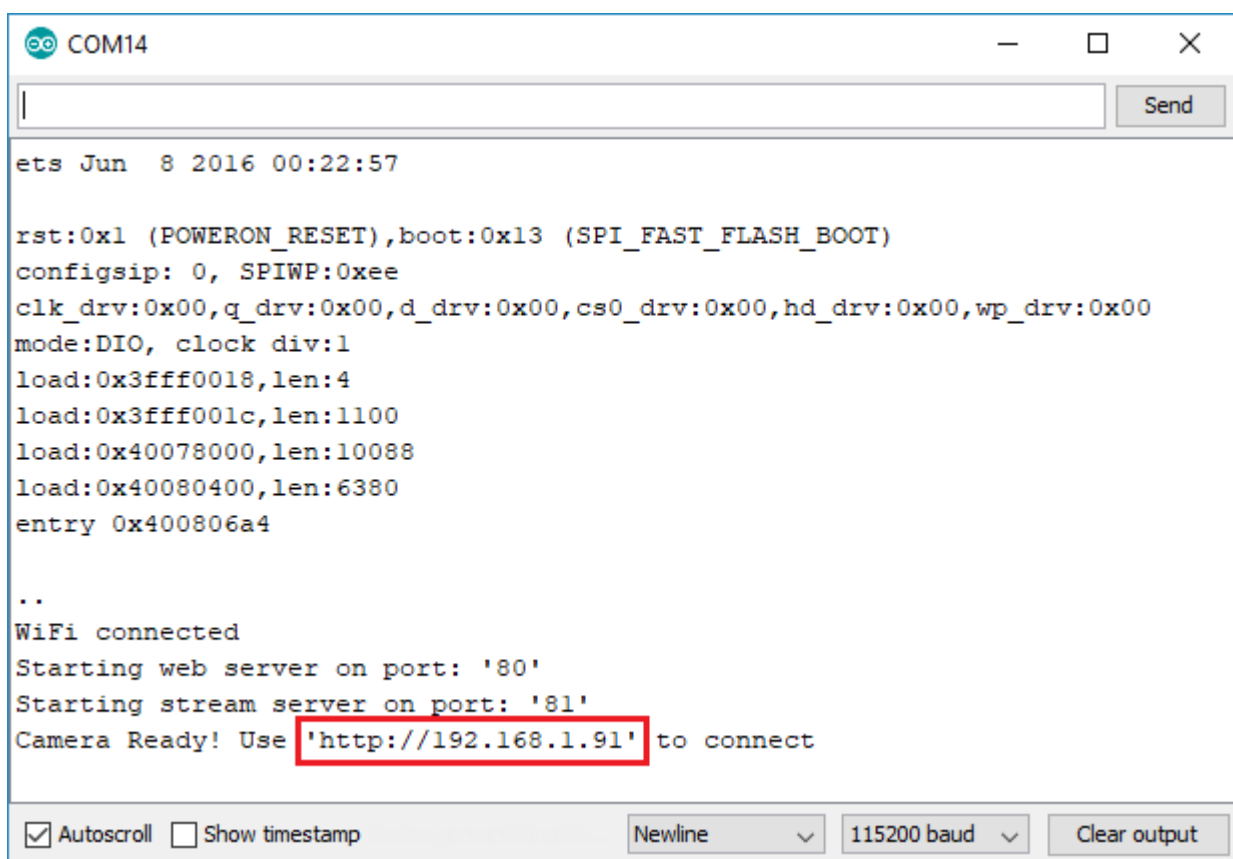
After a few seconds, the code should be successfully uploaded to your board.

Getting the IP address

After uploading the code, disconnect GPIO 0 from GND.

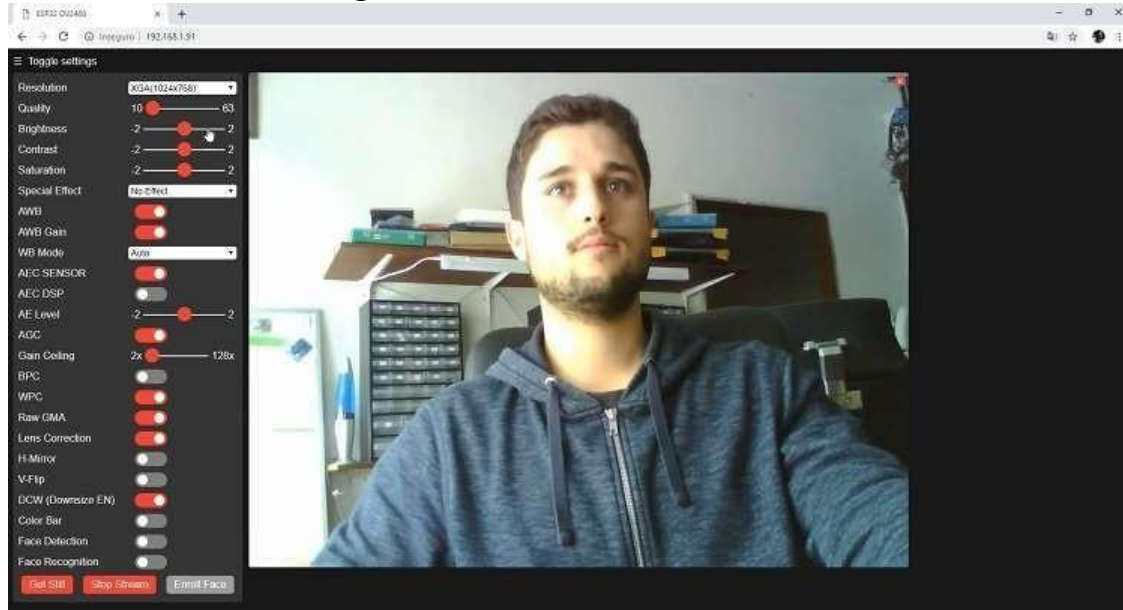
Open the Serial Monitor at a baud rate of 115200. Press the ESP32-CAM on-board Reset button.

The ESP32 IP address should be printed in the Serial Monitor.



Accessing the Video Streaming Server

Now, you can access your camera streaming server on your local network. Open a browser and type the ESP32-CAM IP address. Press the **Start Streaming** button to start video streaming.

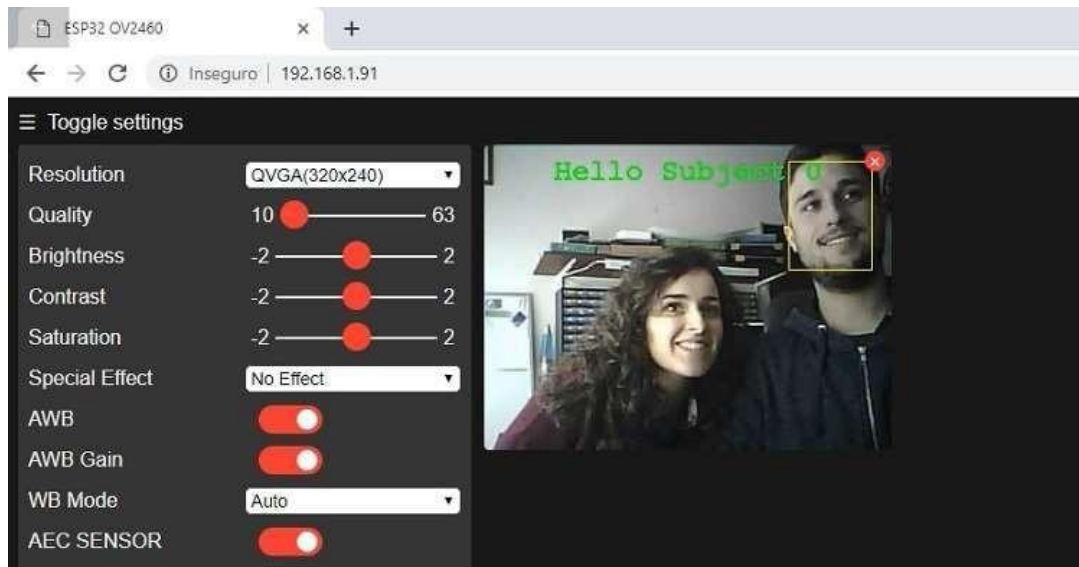


You also have the option to take photos by clicking the **Get Still** button. Unfortunately, this example doesn't save the photos, but you can modify it to use the onboard microSD Card to store the captured photos. There are also several camera settings that you can play with to adjust the image settings.

Finally, you can do face recognition and detection.



First, you need to enroll a new face. It will make several attempts to save the face. After enrolling a new user, it should detect the face later on (subject 0).



And that's it. Now you have your video streaming web server up and running with face detection and recognition with the example from the library.

Troubleshooting

If you're getting any of the following errors, read our **ESP32-CAM Troubleshooting Guide: Most Common Problems Fixed**

- Failed to connect to ESP32: Timed out waiting for packet header
- Camera init failed with error 0x20001 or similar
- Brownout detector or Guru meditation error
- Sketch too big error – Wrong partition scheme selected
- Board at COMX is not available – COM Port Not Selected
- Psram error: GPIO isr service is not installed
- Weak Wi-Fi Signal
- No IP Address in Arduino IDE Serial Monitor
- Can't open webserver
- The image lags/shows lots of latency

IR (Infrared) Sensor

An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects motion. These types of sensors measure infrared radiation, rather than emitting it that which called a passive IR sensor.

Usually, in the infrared spectrum, all the objects radiate some form of thermal radiations. These types of radiations are invisible to our eyes that can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode that is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and these output voltages, change in proportion to the magnitude of the IR light received.

IR Sensor Circuit Diagram and Working Principle

An infrared sensor circuit is one of the basic and popular sensor modules in an electronic device. This sensor is analogous to human's visionary senses which can be used to detect obstacles and it is one of the common applications in real time. This circuit comprises the following components

- LM358 IC 2 IR transmitter and receiver pair
- Resistors of the range of kilo-ohms.
- Variable resistors.
- LED (Light Emitting Diode).

In this project, the transmitter section includes an IR sensor, which transmits continuous IR rays to be received by an IR receiver module. An IR output terminal of the receiver varies depending upon its receiving of IR rays. Since this variation cannot be analyzed as such, therefore this output can be fed to a comparator circuit. Here an operational amplifier (op-amp) of LM 339 is used as a comparator circuit. When the IR receiver does not receive a signal, the potential at the inverting input goes higher than that non-inverting input of the comparator IC (LM339). Thus the output of the comparator goes low, but the LED does not glow. When the IR receiver module receives a signal, the potential at the inverting input goes low. Thus the output of the comparator (LM 339) goes high and the LED starts glowing.

IR Sensor circuit

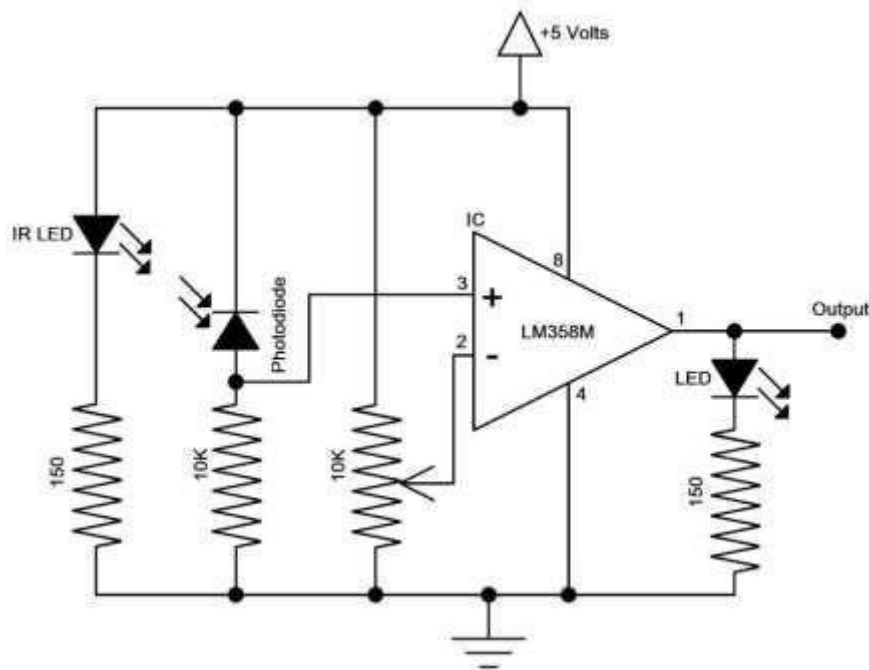


Fig. IR Sensor Circuit diagram

Resistor R1 (100 Ω), R2 (10k Ω), and R3 (330 Ω) are used to ensure that minimum 10 mA current passes through the IR LED Devices like Photodiode and normal LEDs respectively. Resistor VR2 (preset=5k) is used to adjust the output terminals. Resistor VR1 (preset=10k) is used to set the sensitivity of the circuit Diagram. Read more about IR sensors.

Different Types of IR Sensors and Their Applications

IR sensors are classified into different types depending on the applications. The speed sensor is used for synchronizing the speed of multiple motors. The temperature sensor is used for industrial temperature control. PIR sensor is used for automatic door opening system and Ultrasonic sensor is used for distance measurement.

IR Sensor Applications

IR sensors are used in various Sensor based projects and also in various electronic devices which measure the temperature that is discussed in the following.

Radiation Thermometers

IR sensors are used in radiation thermometers to measure the temperature depend on the temperature and the material of the object and these thermometers have some of the following features

- Measurement without direct contact with the object
- Faster response
- Easy pattern measurements

Flame Monitors

These types of devices are used for detecting the light emitted from the flames and to monitor how the flames are burning. The Light emitted from flames extend from UV to IR region types. PbS, PbSe, Two-color detector, pyroelectric detector are some of the commonly employed detectors used in flame monitors.

Moisture Analyzers

Moisture analyzers use wavelengths that are absorbed by the moisture in the IR region. Objects are irradiated with light having these wavelengths (1.1 μm , 1.4 μm , 1.9 μm , and 2.7 μm) and also with reference wavelengths. The Lights reflected from the objects depend upon the moisture content and are detected by an analyzer to measure moisture (ratio of reflected light at these wavelengths to the reflected light at reference wavelength). In GaAs PIN photodiodes, PbS photoconductive detectors are employed in moisture analyzer circuits.

IR Imaging Devices

IR image device is one of the major applications of IR waves, primarily by its property that is not visible. It is used for thermal imagers, night vision devices, etc.



For example Water, rocks, soil, vegetation, and atmosphere, and human tissue all features emit IR radiation. The Thermal infrared detectors measure these radiations in the IR range and map the spatial temperature distributions of the object/area on an image. Thermal imagers usually consist of a Sb (indium antimonite), Gd Hg (mercury-doped germanium), Hg Cd Te (mercury-cadmium-telluride) sensors.

An electronic detector is cooled to low temperatures using liquid helium or liquid nitrogen. Then the Cooling of the detectors ensure that the radiant energy (photons) recorded by the detectors comes from the terrain and not from the ambient temperature of objects within the scanner itself an IR imaging electronic device.

Thus, this is all about IR sensor circuits with working and applications. These sensors are used in many sensor-based electronics projects.

Bob Gear motor

A gear motor is a kind of electrical machinery which can produce high torque at low-speed motor output. These kinds of gear motors are used in many different applications and are typically found in homes and workplaces.

The following are the basic principles in gear motor operation:

Power Source: A gear motor can run on either an alternating current (AC) or direct current (DC) electric motor. The output range of a gear motor can be between 1, 200 and 3, 600 revolutions per minute or RPM. The gearbox operates to reduce the RPM to

produce different torque at the end of the shaft.

Speed Specifications: There are two basic speed specifications for a gearbox, namely the normal speed and the stall speed. These torque specifications essentially assist in the creation of torque. This is done through a series of integrated gears attached to the main shaft and rotor mechanism. The attachment is done through a reduction shaft between the gearbox and the drive shaft. In essence, the gears reduce the speed but increase torque, with the final output in the end.

Applications: A gear motor is often found in many home electrical appliances. These include can openers, control knobs in washer and dryer machines, alarm clocks, and garage door openers. Commercially, a gear motor is often found in motorized hospital beds, heavy-duty cranes, and commercial lifters or jacks.



BOB GEAR MOTOR

Software required:

ARDUINO UNO AND ITS PROGRAMMING

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer. The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

2.1 Overview

The Arduino microcontroller is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The Arduino is open-source, which means hardware is reasonably priced and development software is free. This guide is for students in ME 2011, or students anywhere who are confronting the Arduino for the first time. For advanced Arduino users, prowl the web; there are lots of resources.

This guide covers the Arduino Uno board (Spark fun DEV-09950, \$29.95), a good choice for students and educators. With the Arduino board, you can write programs and create interface circuits to read switches and other sensors, and to

control motors and lights with very little effort. Many of the pictures and drawings in this guide were taken from the documentation on the

This is what the Arduino ide looks like.



The Duemilanove board features an Atmel ATmega328 microcontroller operating at 5 V with 2 Kb of RAM, 32 Kb of flash memory for storing programs and 1 Kb of EEPROM for storing parameters. The clock speed is 16 MHz, which translates to about executing about 300,000 lines of C source code per second. The board has 14 digital I/O pins and 6 analog input pins. There is a USB connector for talking to the host computer and a DC power jack for connecting an external 6-20 V power source, for example a 9 V battery, when running a program while not connected to the host computer. Headers are provided for interfacing to the I/O pins using 22 g solid wire or header connectors.

The Arduino programming language is a simplified version of C/C++. If you know C, programming the Arduino will be familiar. If you do not know C, no need to worry as only a few commands are needed to perform useful functions.

An important feature of the Arduino is that you can create a control program on the host PC, download it to the Arduino and it will run automatically. Remove the USB cable connection to the PC, and the program will still run from the top each time you push the reset button. Remove the battery and put the Arduino board in a closet for six months. When you reconnect the battery, the last program you stored will run. This means that you connect the board to the host PC to

develop and debug your program, but once that is done, you no longer need the PC to run the program.

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

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

Arduino Software (IDE)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches

Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension **.ino**. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.



Verify

Checks your code for errors compiling it.



Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"



New

Creates a new sketch.



Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the **File | Sketchbook** menu instead.



Save

Saves your sketch.



Serial

Monitor

Opens the serial monitor.

Additional commands are found within the five menus: **File**, **Edit**, **Sketch**, **Tools**, **Help**. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

File

- ***New***

Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

- *Open*
Allows to load a sketch file browsing through the computer drives and folders.
- *Open* *Recent*
Provides a short list of the most recent sketches, ready to be opened.
- *Sketchbook*
Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.
- *Examples*
Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- *Close*
Closes the instance of the Arduino Software from which it is clicked.
- *Save*
Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.
- *Save* *as...*
Allows to save the current sketch with a different name.
- *Page* *Setup*
It shows the Page Setup window for printing.
- *Print*
Sends the current sketch to the printer according to the settings defined in Page Setup.
- *Preferences*
Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- *Quit*
Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

Edit

- *Undo/Redo*
Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- *Cut*
Removes the selected text from the editor and places it into the clipboard.

- *Copy*
Duplicates the selected text in the editor and places it into the clipboard.
- *Copy* *for* *Forum*
Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- *Copy* *as* *HTML*
Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- *Paste*
Puts the contents of the clipboard at the cursor position, in the editor.
- *Select* *All*
Selects and highlights the whole content of the editor.
- *Comment/Uncomment*
Puts or removes the // comment marker at the beginning of each selected line.
- *Increase/Decrease* *Indent*
Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- *Find*
Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- *Find* *Next*
Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- *Find* *Previous*
Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Sketch

- *Verify/Compile*
Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- *Upload*
Compiles and loads the binary file onto the configured board through the configured Port.
- *Upload* *Using* *Programmer*
This will overwrite the bootloader on the board; you will need to use Tools >

Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a *Tools -> Burn Bootloader* command must be executed.

- *Export* *Compiled* *Binary*
Saves a .hex file that may be kept as archive or sent to the board using other tools.
- *Show* *Sketch* *Folder*
Opens the current sketch folder.
- *Include* *Library*
Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see [libraries](#) below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- *Add* *File...*
Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side of the toolbar.

Tools

- *Auto* *Format*
This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- *Archive* *Sketch*
Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.
- *Fix* *Encoding* *&* *Reload*
Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- *Serial* *Monitor*
Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.
- *Board*
Select the board that you're using. See below for [descriptions of the various boards](#).

- *Port*

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

- *Programmer*

For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

- *Burn*

Bootloader

The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the **Boards** menu before burning the bootloader on the target board. This command also set the right fuses.

Help

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

- *Find*

in

Reference

This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the **File > Sketchbook** menu or from the **Open** button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the **Preferences** dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in

version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading

Before uploading your sketch, you need to select the correct items from the **Tools** > **Board** and **Tools** > **Port** menus. The boards are described below. On the Mac, the serial port is probably something like **/dev/tty.usbmodem241** (for an Uno or Mega2560 or Leonardo) or **/dev/tty.usbserial-1B1** (for a Duemilanove or earlier USB board), or **/dev/tty.USA19QW1b1P1.1** (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be **/dev/ttyACMx** , **/dev/ttyUSBx** or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the **Upload** item from the **Sketch** menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino **bootloader**, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from

the **Sketch** > **Import Library** menu. This will insert one or more **#include** statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its **#include** statements from the top of your code.

There is a [list of libraries](#) in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these [instructions for installing a third-party library](#).

To write your own library, see [this tutorial](#).

Third-Party Hardware

Support for third-party hardware can be added to the **hardware** directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the **hardware** directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the [Arduino IDE 1.5 3rd party Hardware specification](#).

Serial Monitor

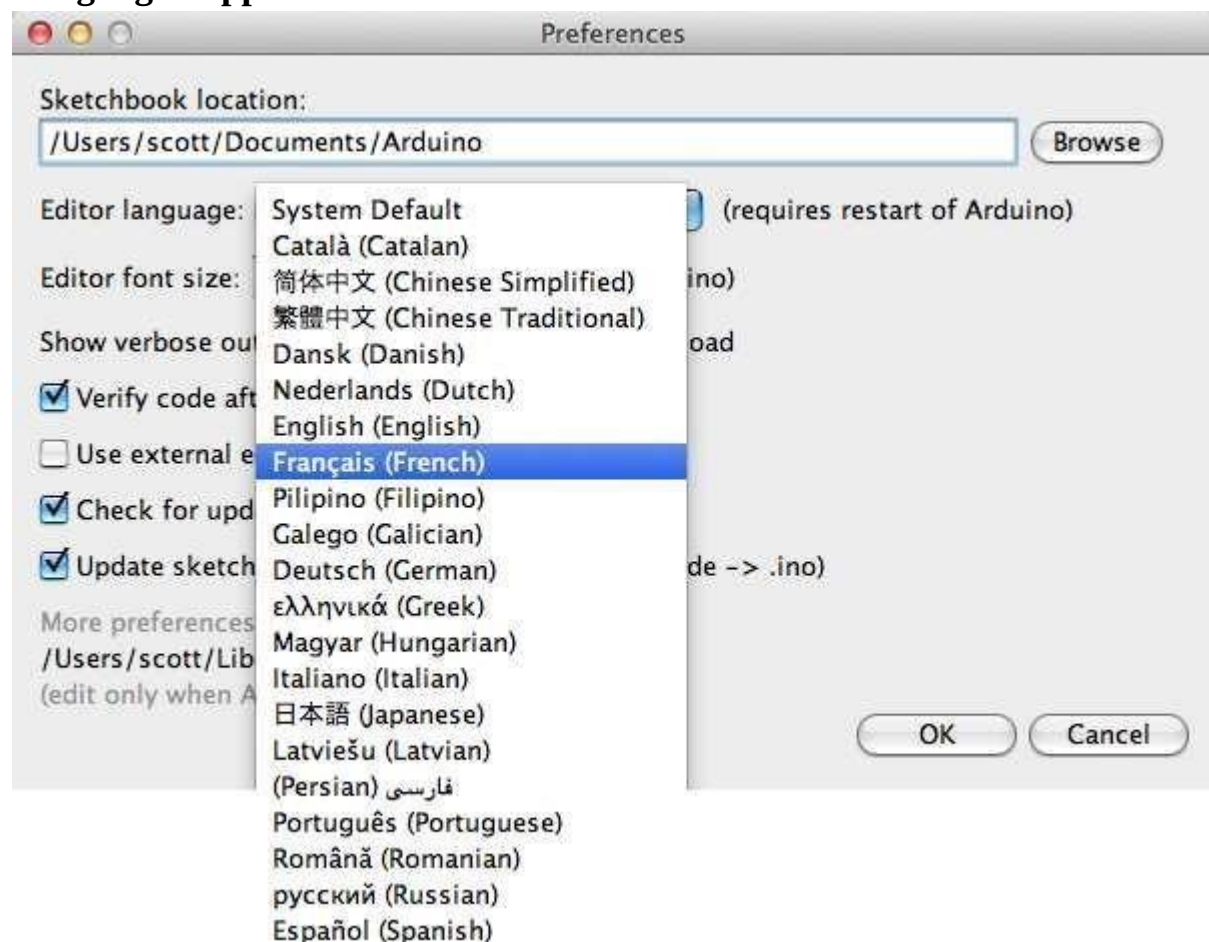
This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to **Serial.begin** in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the [interfacing page](#) for details).

Preferences

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

Language Support



Since version 1.0.1 , the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

If you would like to change the language manually, start the Arduino Software (IDE) and open the **Preferences** window. Next to the **Editor Language** there is a dropdown menu of currently supported languages. Select your preferred

language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting **System Default** from the **Editor Language** drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

Boards

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards [here](#).

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The [Boards Manager](#) included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

- *Arduino* *Yùn*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino* *Uno*
An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino* *Diecimila* *or* *Duemilanove* *w/ ATmega168*
An ATmega168 running at 16 MHz with auto-reset.
- *Arduino* *Nano* *w/ ATmega328P*
An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.
- *Arduino/Genuino* *Mega* *2560*
An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- *Arduino* *Mega*
An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino* *Mega* *ADK*
An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino* *Leonardo*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino* *Micro*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino* *Esplora*
An ATmega32u4 running at 16 MHz with auto-reset.
- *Arduino* *Mini* *w/ ATmega328P*
An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino* *Ethernet*
Equivalent to Arduino UNO with an Ethernet shield: An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino* *Fio*
An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino* *BT* *w/ ATmega328P*
ATmega328P running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM..
- *LilyPad Arduino* *USB*
An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
- *LilyPad Arduino*
An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P*
An ATmega328P running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328P; 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino NG or older w/ ATmega168*
An ATmega168 running at 16 MHz *without* auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Robot Control*
An ATmega328P running at 16 MHz with auto-reset.
- *Arduino Robot Motor*
An ATmega328P running at 16 MHz with auto-reset.
- *Arduino Gemma*
An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

For instructions on installing support for other boards, see [third-party hardware](#) above.

Last revision 2015/09/07 by SM

The text of the Arduino getting started guide is licensed under a [Creative Commons Attribution-ShareAlike 3.0 License](#). Code samples in the guide are released into the public domain.

Blynk app:

What do I need to Blynk?

At this point you might be thinking: “Ok, I want it. What do I need to get started?” – Just a couple of things, really:

1. Hardware.

An Arduino, Raspberry Pi, or a similar development kit.

Blynk works over the Internet. This means that the hardware you choose should be able to connect to the internet. Some of the boards, like Arduino Uno will need an Ethernet or Wi-Fi Shield to communicate, others are already Internet-enabled: like the ESP8266, Raspberri Pi with WiFi dongle, Particle Photon or SparkFun Blynk Board. But even if you don't have a shield, you can connect it over USB to your laptop or desktop (it's a bit more complicated for newbies, but we got you covered). What's cool, is that the [list of hardware](#) that works with Blynk is huge and will keep on growing.

2. A Smartphone.

The Blynk App is a well-designed interface builder. It works on both iOS and Android, so no holy wars here, ok?

Downloads

Blynk Apps for iOS or Android



Blynk Library

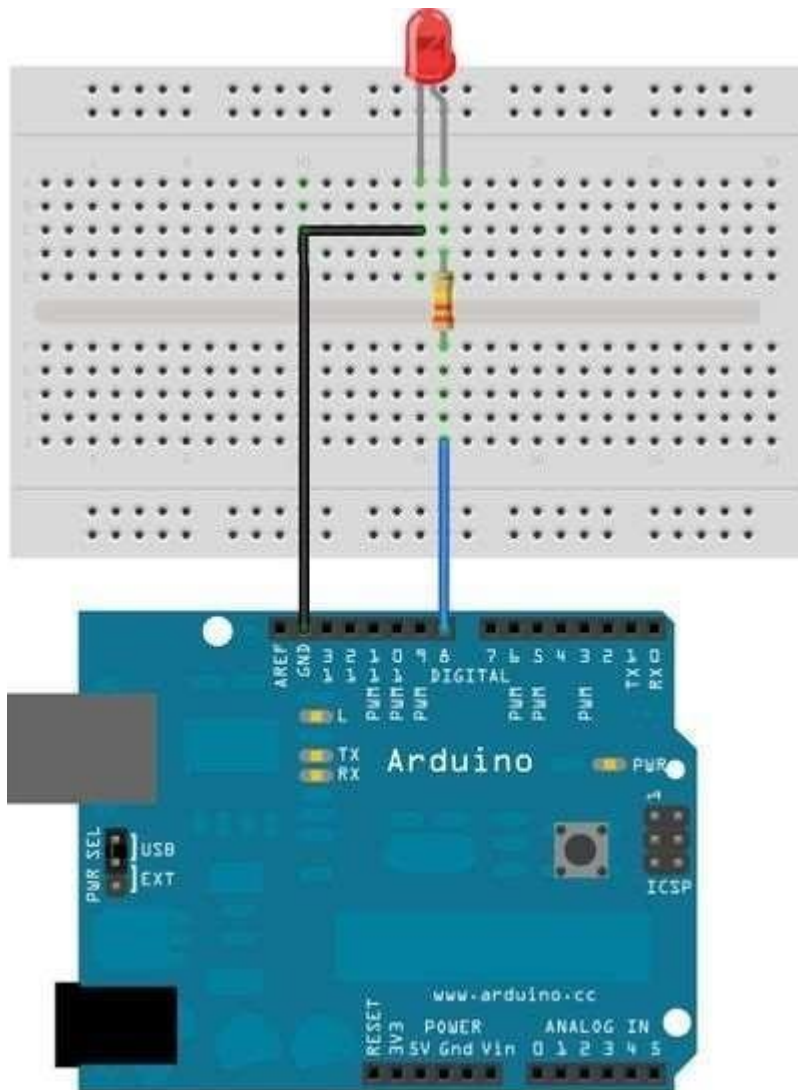
DOWNLOAD THE BLYNK LIBRARY

In case you forgot, or don't know how to install Arduino libraries [click here](#).

Getting Started

Let's get you started in 5 minutes (reading doesn't count!). We will switch on an LED connected to your Arduino using the Blynk App on your smartphone.

Connect an LED as shown here:

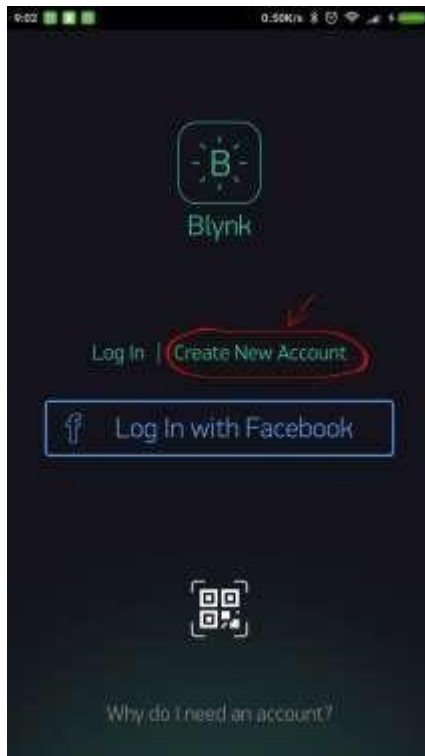


Getting Started With The Blynk App

1. Create a Blynk Account

After you download the Blynk App, you'll need to create a New Blynk account. This account is separate from the accounts used for the Blynk Forums, in case you already have one.

We recommend using a **real** email address because it will simplify things later.



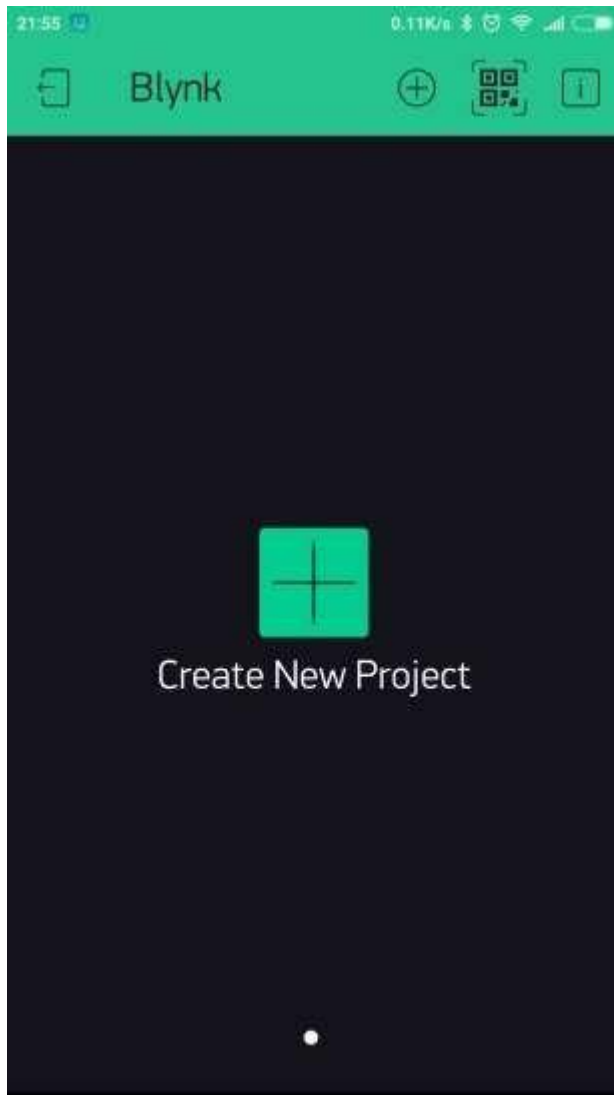
Why do I need to create an account?

An account is needed to save your projects and have access to them from multiple devices from anywhere. It's also a security measure.

You can always set up your own Private Blynk Server and have full control.

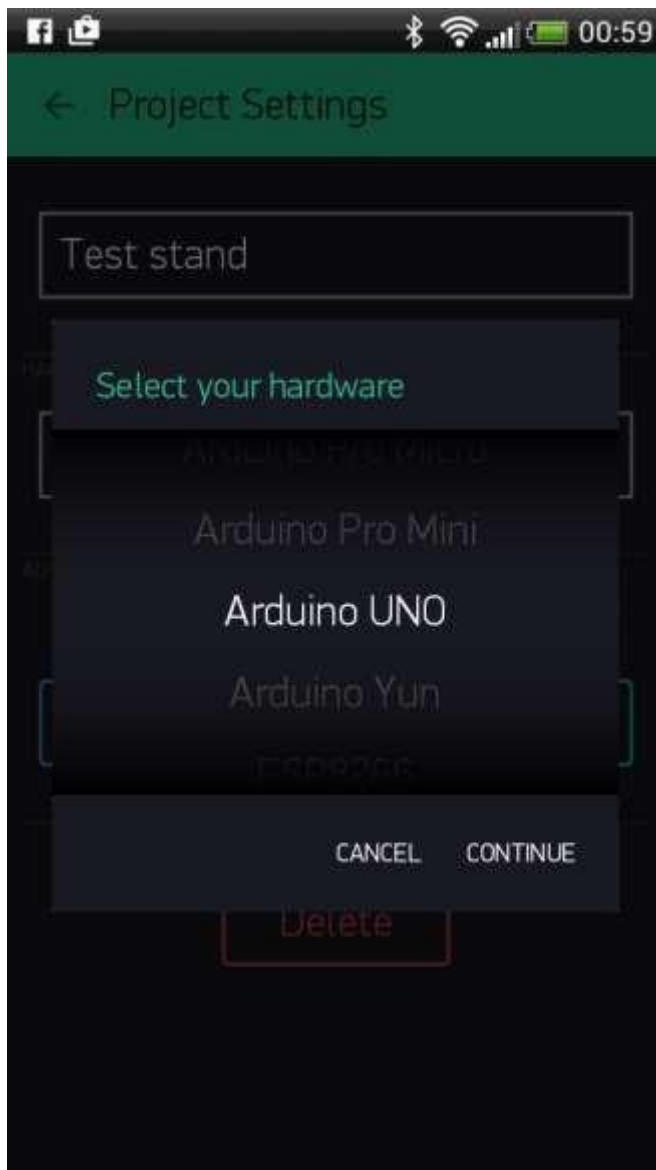
2. Create a New Project

After you've successfully logged into your account, start by creating a new project.



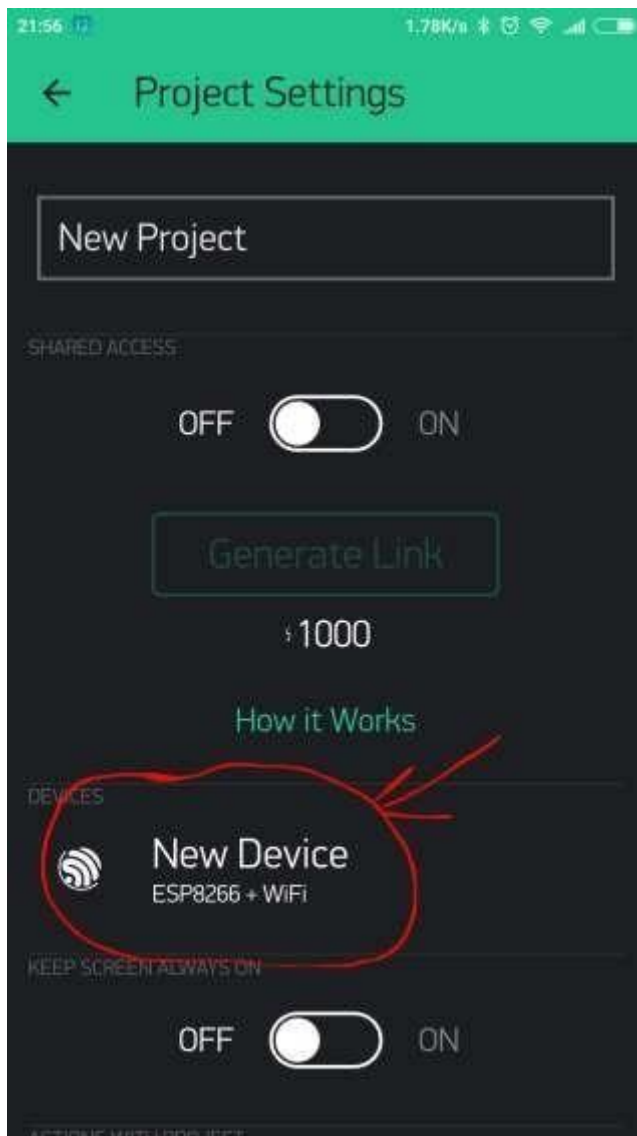
3. Choose Your Hardware

Select the hardware model you will use. Check out the [list of supported hardware!](#)



4. Auth Token

Auth Token is a unique identifier which is needed to connect your hardware to your smartphone. Every new project you create will have its own Auth Token. You'll get Auth Token automatically on your email after project creation. You can also copy it manually. Click on devices section and selected required device :



And you'll see token :

13:48 0.02K/s

← My Devices

Device Name

HARDWARE MODEL

ESP8266

CONNECTION TYPE

WiFi

AUTH TOKEN

f064daa6f6a7411ebe591f796f2e6a17

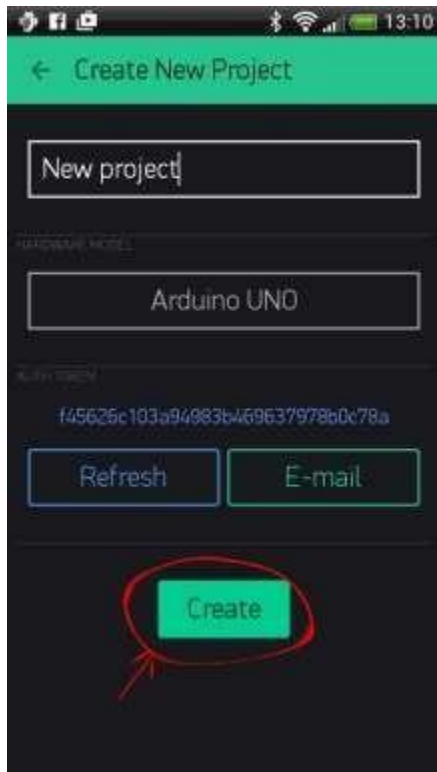
Refresh E-mail

Create

NOTE: Don't share your Auth Token with anyone, unless you want someone to have access to your hardware.

It's very convenient to send it over e-mail. Press the e-mail button and the token will be sent to the e-mail address you used for registration. You can also tap on the Token line and it will be copied to the clipboard.

Now press the **"Create"** button.

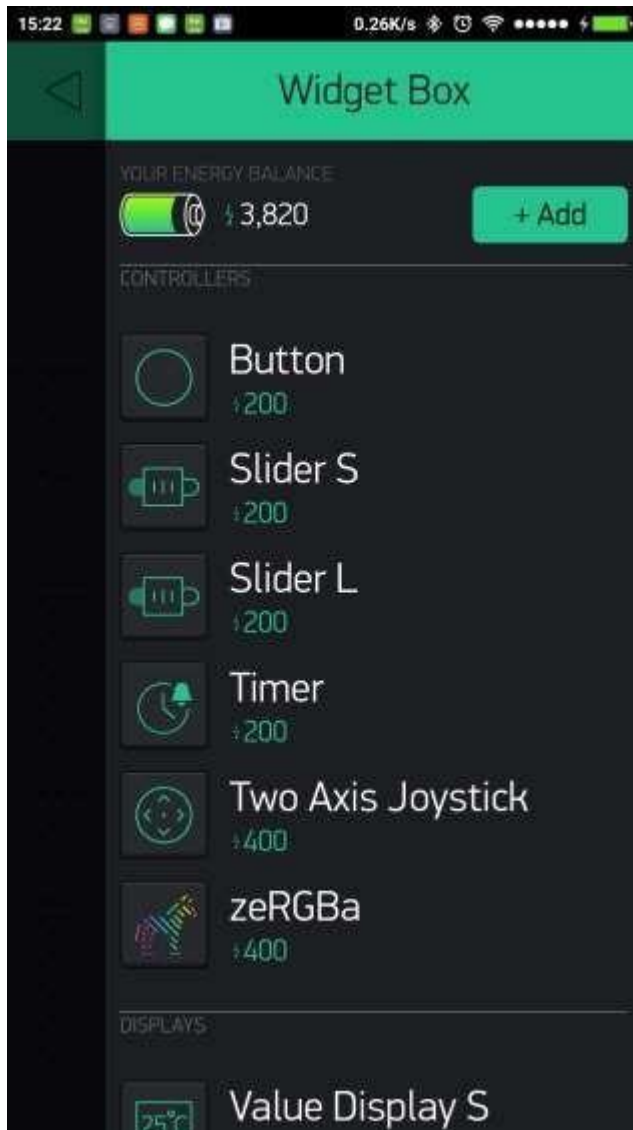


5. Add a Widget

Your project canvas is empty, let's add a button to control our LED.

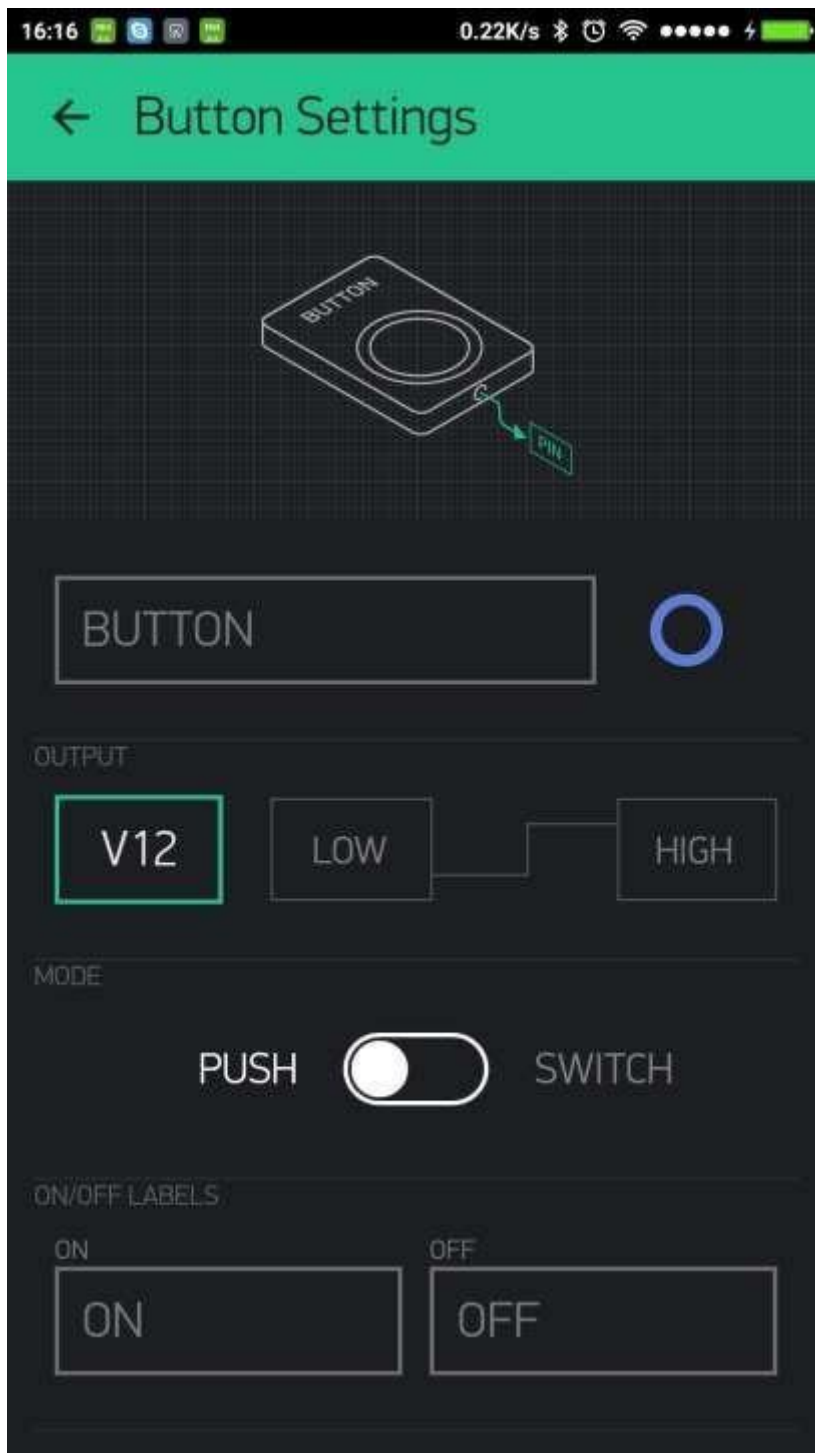
Tap anywhere on the canvas to open the widget box. All the available widgets are located here. Now pick a button.

Widget Box

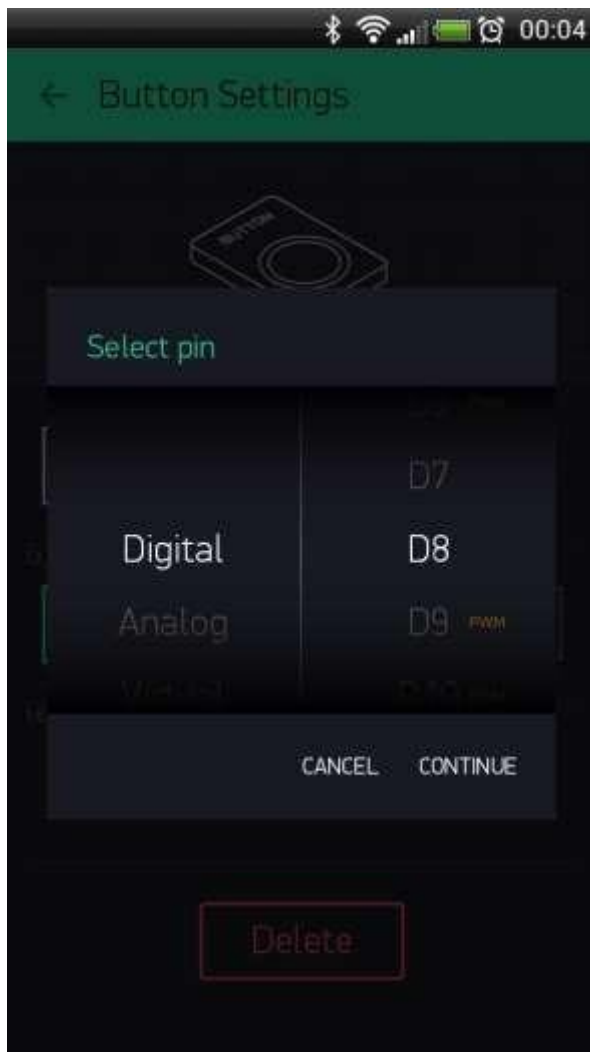


Drag-n-Drop - Tap and hold the Widget to drag it to the new position.

Widget Settings - Each Widget has it's own settings. Tap on the widget to get to them.



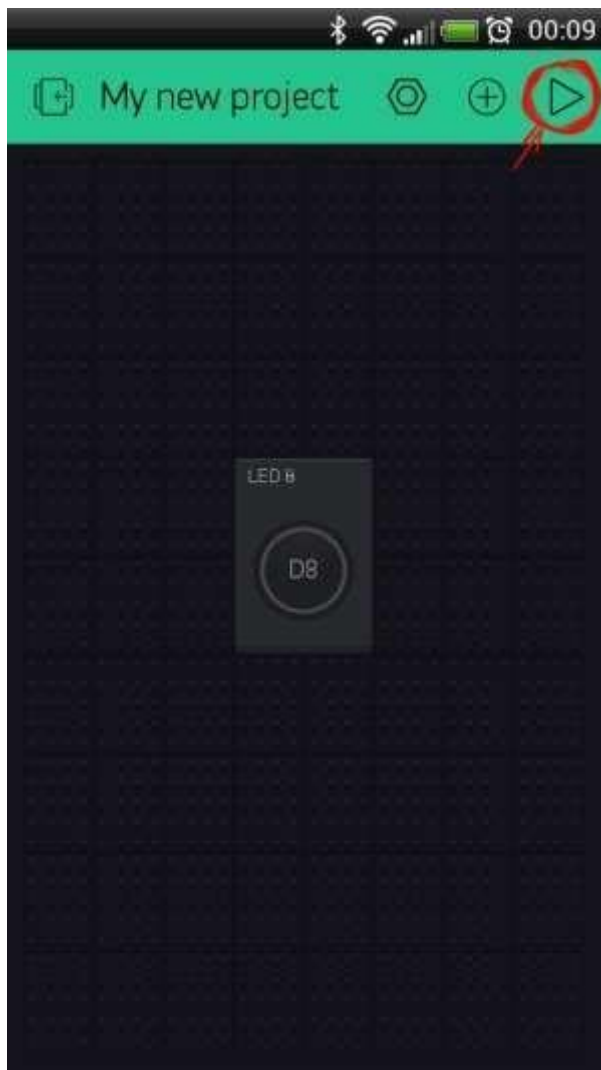
The most important parameter to set is **PIN** . The list of pins reflects physical pins defined by your hardware. If your LED is connected to Digital Pin 8 - then select **D8** (**D** - stands for **D**igital).



6. Run The Project

When you are done with the Settings - press the **PLAY** button. This will switch you from EDIT mode to PLAY mode where you can interact with the hardware. While in PLAY mode, you won't be able to drag or set up new widgets, press **STOP** and get back to EDIT mode.

You will get a message saying "Arduino UNO is offline". We'll deal with that in the next section.



Getting Started With Hardware

How To Use an Example Sketch

You should by now have the Blynk Library installed on your computer. If not - [click here](#).

Example sketches will help you get your hardware online quickly and major Blynk features.

Open the example sketch according to the hardware model or shield you are using.



Let's take a look at the example sketch for an Arduino UNO + Ethernet shield

```
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <Ethernet.h>
#include <BlynkSimpleEthernet.h>
```

```
char auth[] = "YourAuthToken";
```

```
void setup()
{
  Serial.begin(9600); // See the connection status in Serial Monitor
  Blynk.begin(auth); // Here your Arduino connects to the Blynk Cloud.
}
```

```
void loop()
{
  Blynk.run(); // All the Blynk Magic happens here...
}
```

Auth Token

In this example sketch, find this line:

```
char auth[] = "YourAuthToken";
```

This is the Auth Token that you emailed yourself. Please check your email and copy it, then paste it inside the quotation marks.

It should look similar to this:

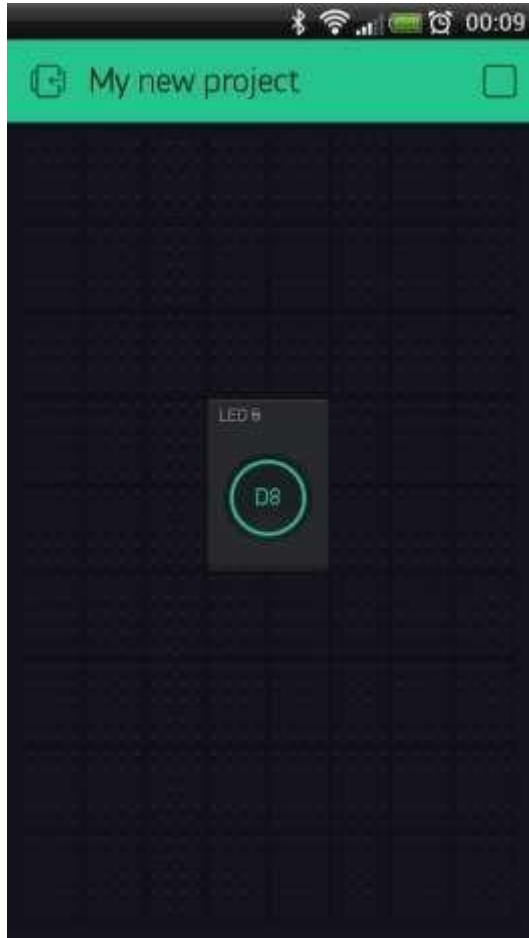
```
char auth[] = "f45626c103a94983b469637978b0c78a";
```

Upload the sketch to the board and open Serial Terminal. Wait until you see something like this:

```
Blynk v.X.X.X
Your IP is 192.168.0.11
Connecting...
Blynk connected!
```

**Congrats! You are all set! Now your hardware is connected to the Blynk Cloud!
Blynking**

Go back to the Blynk App, push the button and turn the LED on and off! It should be Blynking.



Check out [other example sketches](#).

Feel free to experiment and combine different examples together to create your own amazing projects.

For example, to attach an LED to a PWM-enabled Pin on your Arduino, set the slider widget to control the brightness of an LED. Just use the same steps described above.

Hardware set-ups

Arduino over USB (no shield)

If you don't have any shield and your hardware doesn't have any connectivity, you can still use Blynk – directly over USB :

1. Open [Arduino Serial USB example](#) and change [Auth Token](#)
2. `// You could use a spare Hardware Serial on boards that have it (like Mega)`
3. `#include <SoftwareSerial.h>`
4. `SoftwareSerial DebugSerial(2, 3); // RX, TX`

```

5.
6. #define BLYNK_PRINT DebugSerial
7. #include <BlynkSimpleStream.h>
8.
9. // You should get Auth Token in the Blynk App.
10. // Go to the Project Settings (nut icon).
11. char auth[] = "YourAuthToken";
12.
13. void setup()
14. {
15.     // Debug console
16.     DebugSerial.begin(9600);
17.
18.     // Blynk will work through Serial
19.     Serial.begin(9600);
20.     Blynk.begin(auth, Serial);
21. }
22.
23. void loop()
24. {
25.     Blynk.run();
26. }

```

26. Run the script which is usually located in /scripts folder:

- Windows: My Documents\Arduino\libraries\Blynk\scripts
- Mac User\$/Documents/Arduino/libraries/Blynk/scripts

On Windows:

Open cmd.exe

Write your path to blynk-ser.bat folder. For example:

```
cd C:\blynk-library-0.3.1\blynk-library-0.3.1\scripts
```

Run blynk-ser.bat file. For example : blynk-ser.bat -c COM4 (where COM4 is port with your Arduino)

And press “Enter”, press “Enter” and press “Enter”

On Linux and Mac:

Navigate to /scripts folder. For example:

```
cd User$/Documents/Arduino/libraries/Blynk/scripts
```

When inside this folder, run:

```
user:scripts User$ ./blynk-ser.sh
```

Warning: Do not close terminal window with running script.

In some cases you may also need to perform :

```
user:scripts User$ chmod +x blynk-ser.sh
```

You may need also to run it with `sudo`
user:scripts User\$ `sudo ./blynk-ser.sh`

This is what you'll see in Terminal app on Mac (usbmodem address can be different):

```
[ Press Ctrl+C to exit ]
```

```
/dev/tty.usbmodem not found.
```

```
Select serial port [ /dev/tty.usbmodem1451 ]:
```

Copy the serial port address: `/dev/tty.usbmodem1451` and paste it back:

```
Select serial port [ /dev/tty.usbmodem1451 ]: /dev/tty.usbmodem1451
```

After you press Enter, you should see an output similar to this:

```
Resetting device /dev/tty.usbmodem1451...
```

```
Connecting:
```

```
GOPEN:/dev/tty.usbmodem1451,raw,echo=0,cllocal=1,cs8,nonblock=1,ixoff=0,ixon=0,ispeed=9600,ospeed=9600,crtscts=0 <-> openssl-connect:blynk-cloud.com:9443,cafile=/Users/.../server.crt,nodelay
```

```
2015/10/03 00:29:45 socat[30438.2046857984] N opening character device "/dev/tty.usbmodem1451" for reading and writing
```

```
2015/10/03 00:29:45 socat[30438.2046857984] N opening connection to
```

```
LEN=16 AF=2 45.55.195.102:9443
```

```
2015/10/03 00:29:45 socat[30438.2046857984] N successfully connected from local address LEN=16 AF=2 192.168.0.2:56821
```

```
2015/10/03 00:29:45 socat[30438.2046857984] N SSL connection using AES128-SHA
```

```
2015/10/03 00:29:45 socat[30438.2046857984] N starting data transfer loop with FDs [3,3] and [4,4]
```

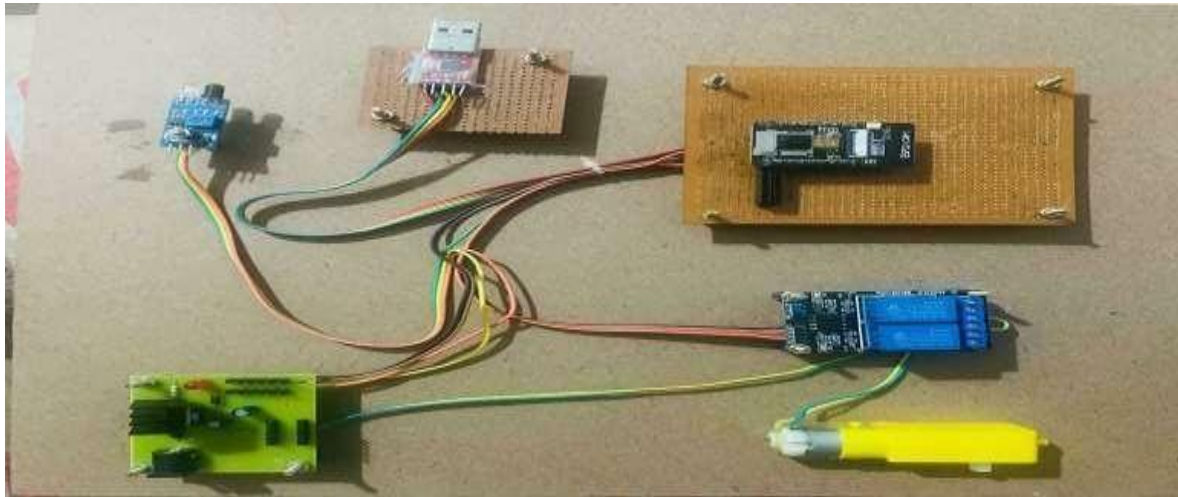
NOTE: Arduino IDE may complain with “programmer is not responding”. You need to terminate script before uploading new sketch.

Conclusion:

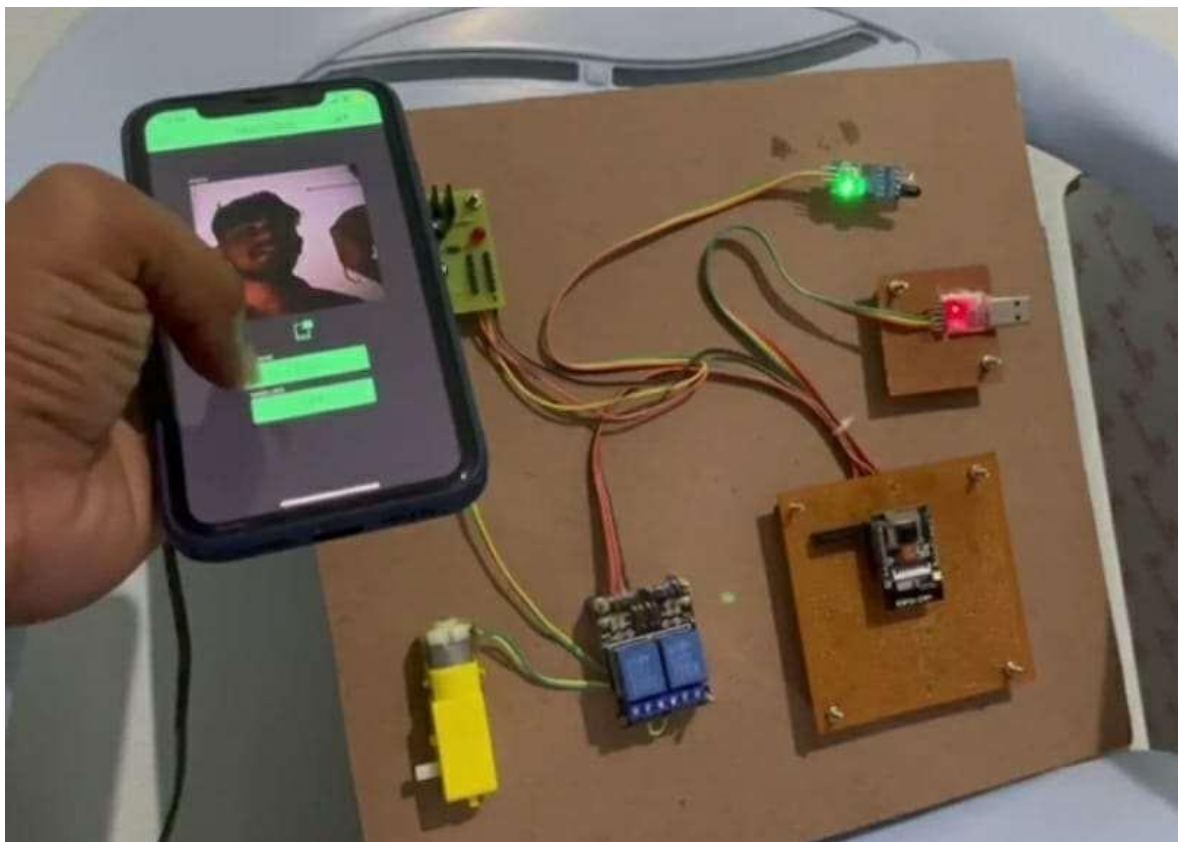
By this Project we can lock the Door automatically by Using Blink app setup and it plays an Important role to Detect the Persons those who are familiar with Us and we can operate this lock from any where in the world.

Result:

Project setup



Working of Intelligent Door Lock system Using IOT



References:

- [1] Jie-Ci Yang et. all An Intelligent Automated Door Control System Based on a Smart Camera.

- [2] S. Nazeem Basha et all An Intelligent Door system using Raspberry pi and Amazon Web services IOT.

- [3] Richard Grimmett, Raspberry Pi Robotic Projects. Packt Publishing.

- [4] W. F. Abaya, J. Basa, M. Sy, A. C. Abad and E. P. Dadios, "Low cost smart security camera with night vision capability using Raspberry Pi and OpenCV," 2019 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), Palawan, 2020, pp. 1-6

- [5] Priya Pasumarti¹, P. Purna Sekhar "Classroom Attendance Using Face Detection and Raspberry-Pi" International Research journal of Engineering and Technology (IRJET), Volume: 05 Issue: 03. p3-p5,Mar-2021

- [6] S Rajkumar, J Prakash, "Automated attendance using Raspberry pi", International Journal Of Pharmacy & Technology (IJPT), Vol. 8, No. 3, pp. 16214-16221, September 2020.