ECE 110 Honors Lab Final Report

DoorGuardian:

An Autonomous Remote Unlocking System

University of Illinois, Urbana-Champaign

Vayun Gupta - vayung2 (ECE 110)

Nippun Sabharwal - nippuns2 (ECE 110)

Siddarth Natarajan - sn28 (ECE 110)

Table of Contents

Abstract:

The idea for this project stemmed from our everyday experience having to use an ID Card to unlock doors across campus. Apart from the obvious hassles of using such a system, it separates the identity of a person from their innate self and assigns it to a piece of plastic. We believe that widespread distribution of a convenient way for authentication offers a novel opportunity to democratize smart lock technology and make transit simpler and faster.

Introduction

The aim of this project is to develop a remote autonomous unlocking system which can be used in order to open a lock remotely. The system uses an Arduino microcontroller in order to integrate all of the components for this project. An OV7670 camera module is utilized in order to send the image of the person in front of a door to the owner of the house. The owner of the house can see these images on their personal laptops and act on it by sending a message to a telegram bot we developed. The telegram bot sends a signal to an ESP32 Bluetooth microcontroller which is used in order to switch on the SG90 servo motors. In this case, the servo motors act as the switching mechanism for our door lock.

Through this project we wanted to learn the mechanisms of Automation, cyber-security, Internet of Things, Networking, APIs, and App development. We see a lot of potential in this technology and intend to explore other applications of such a device. We wanted to use this lab section as a learning process so that we can explore different types of hardware devices and microcontrollers, as well as the development of chat bots in relation to hardware-software integration.

We didn't just want to build a project that is valuable in theory but is not feasible practically. The most feasible application of our product is for Airbnb owners, who can now remotely and securely let new guests into their Airbnb every few days, rather than the current unsecure practice of leaving keys under the doormat. With the *DoorGuardian*, one can now get faster access into buildings, since verification is low latency (takes just a few seconds). Homeowners on vacation can let their trusted friends into their homes for maintenance. Thus, this device is a viable solution that can be used by users around the world.

With just the addition of pre-trained CV software on a dataset of UI student images, the University can provide hassle-free entry with the need of ID Cards or complex passwords. This is especially useful in improving accessibility for persons with disabilities, especially physical disabilities such as limb impairment, as one can just rely on a picture to unlock. Such a system can also be easily updated with records of recognized faces by inserting JPEGs into the system, with the coming of the new academic year.

The following are features of the *DoorGuardian* system:

- Works with a phone
- Motorized Door Unlocking
- Enhanced door security
- Automated log of entries and exits from a place.
- Upgradable to better iterations when needed.
- Camera with the ability to give live feed playback.

2. Background Research

Ultrasonic Sensor:

The HC-SR04 Ultrasonic Distance Sensor emits an ultrasound at 40 000 Hz which travels through the air. If there is an object or obstacle on its path, it will bounce back to the module. Using the travel time and the speed of the sound, it can calculate the distance as follows:

$$Distance\ bw\ Sensor\ and\ Obstacle = \frac{Time(ultrasound\ to\ return\ to\ module) \times Speed\ of\ Sound}{2}$$

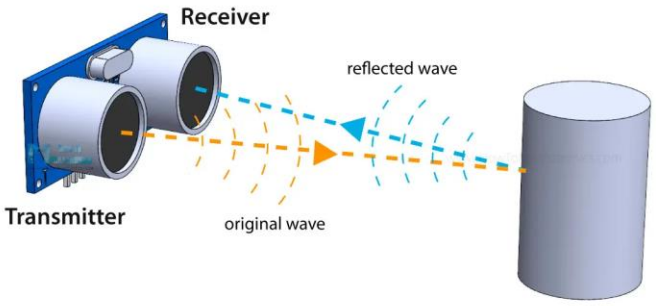


Fig 1: Ultrasonic Sensor detecting an obstacle.

APIs:

API stands for Application Programming Interface. APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone.

The steps to implement a new API include:

1. Obtaining an API key. This is done by creating a verified account with the API provider.
2. Set up an HTTP API client. This tool allows you to structure API requests easily using the API keys received.
3. If you don't have an API client, you can try to structure the request yourself in your browser by referring to the API documentation.
4. Once you are comfortable with the new API syntax, you can start using it in your code.

UART Protocol:

UART, or Universal Asynchronous Receiver-Transmitter, is one of the most used device-to-device communication protocols. UART is a hardware communication protocol that uses asynchronous serial communication with configurable speed. Asynchronous means there is no clock signal to synchronize the output bits from the transmitting device going to the receiving end.

The two signals of each UART device are named:

- Transmitter (Tx)
- Receiver (Rx)

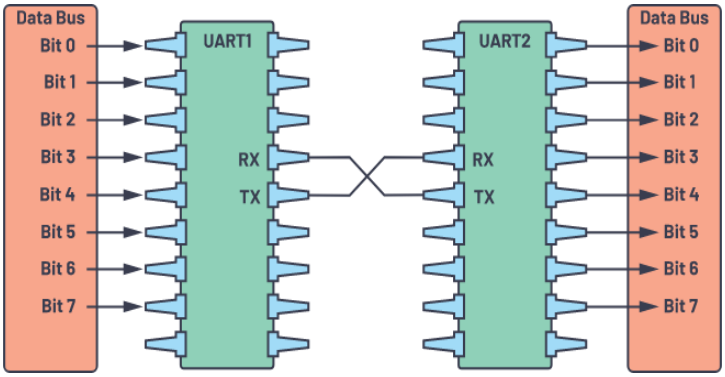


Fig 2 : Two Data Buses directly communicate with each other using UART.

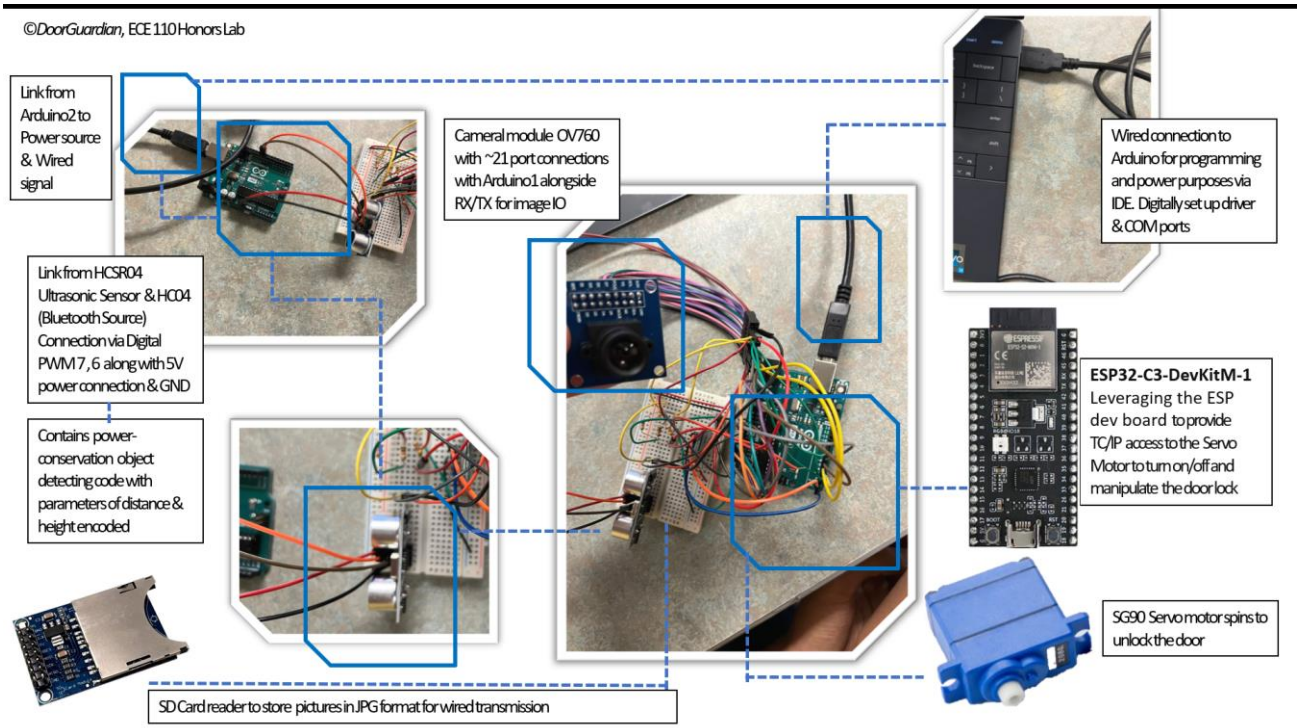
The main purpose of a transmitter and receiver line for each device is to transmit and receive serial data intended for serial communication.

In this project, the ESP32, which connects to the internet and the telegram bot, communicates with the Arduino Uno via UART. Furthermore, the Camera Module also communicates with the Arduino Uno via UART.

3. **Project Outline**

The following parts are used in making the Autonomous Remote unlocking system:

1. The Ultrasonic sensor: detects an approaching person and turns on the Arduino and the camera. This saves power from having to keep the camera on 24 × 7, even when there is no one around.



2. The Camera: captures images of this person to the user for approval.

3. The Arduino: The Arduino IDE uses a software library and UART to connect with the camera. It sends the image to the telegram bot via a laptop and an automation system.

4. The Automation System: takes the image from the Arduino serial monitor and automatically uploads it to the telegram bot.

5. The ESP32: The ESP32 connects to the internet and receives the door open and door close commands from the telegram bot.

6. The Telegram Bot: The telegram bot, which we developed ourselves with Telegram's API on the Arduino IDE, acts as the interface between the Arduino microcontroller and the motor based on user input. The user receives images like they would from a friend in a telegram conversation, and they can open the door simply by messaging the bot "door open" and close the door with "door close".

7. The Motors and Motor Driver: The ESP32 receives a signal from the telegram bot and gives a digital high output to the motor driver, which starts the motor which opens/closes the door.

4. Design

Part 1: Sending the Images to Telegram

4.1.1 Ultrasonic Sensor

The Ultrasonic sensor is used to measure the distance of the person from the camera and then sends a signal to the Arduino when the person is close enough to the door so as to activate the camera module for user identification,

This is a particularly useful feature since it helps us make a Power-Conserving System and only turn on the camera module as and when needed. The system is usually in a low-power standby state which gets activated when it meets the parameters, we put in place to ensure that it is a human that is passing by. For our project, we have set the distance threshold to be 50 centimeters. That is, the camera module will start taking pictures of the person only when he/she is <50 centimeters from the door. Another constraint that minimizes false positives is that the ultrasonic will be placed at a height (adjustable) of 5 feet so that pets/environment does not activate the system.

However, one major challenge associated with the ultrasound sensor is that it is not viable for distances over a certain maximum, in our case, this is around 250.

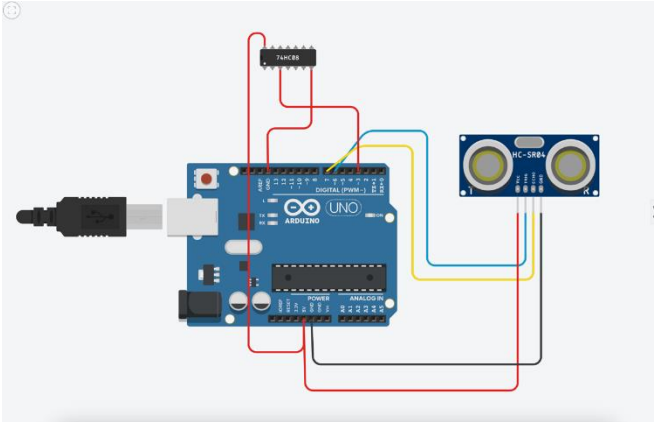


Fig 3: Ultrasound Sensor - Arduino Connections

The code that we developed in order to connect the Arduino with the Ultrasound sensor is as follows:

```
// defines pins numbers
const int trigPin = 9;
const int echoPin = 10;
// defines variables
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
}
void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance, threshold is 50 cm.
  distance = duration * 0.034 / 2;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);
}
```

The operating voltage of our ultrasonic sensor is 5V and can detect obstacles up to a maximum of 157’’, although it is unreliable at such large distances. It draws about 5mA of current. The sensor has 2 main pins: a trigger pin and an echo pin. These pins are used in order to send a pulse to the obstacle and then the signal bounces from the object and enters back into the detector. The process of distance detection takes around 25 milliseconds.

4.1.2 OV7670 Camera Module

The camera module connects with the Arduino microcontrollers and transmits signals using the UART protocol (RX-TX). The RX pin on the Arduino receives image data in the form of an encoded signal from the camera module and the TX pin retransmits certain data back to the camera.

The pins on the camera module send the image data on the Arduino and using the ArduinoImageCapture extension, we were able to extract and slowly the brightness of the image develops when the camera is exposed to the image of the user. This image, once viewed by the owner of the house, can act on it and move on to the next stage of the process (Explained further in the document).

The camera is powered by a voltage of 3.3V and generates an image of 0.3 megapixels. The reason associated with the time delay with image generation is related to the XCLK, PCLK and TCLK pins on the camera, which generate an image upon the clock cycle of the camera.

Pin	Type	Description
VDD**	Supply	Power supply
GND	Supply	Ground level
SDIOC	Input	SCCB clock
SDIOD	Input/Output	SCCB data
VSYNC	Output	Vertical synchronization
HREF	Output	Horizontal synchronization
PCLK	Output	Pixel clock
XCLK	Input	System clock
D0-D7	Output	Video parallel output
RESET	Input	Reset (Active low)
PWDN	Input	Power down (Active high)

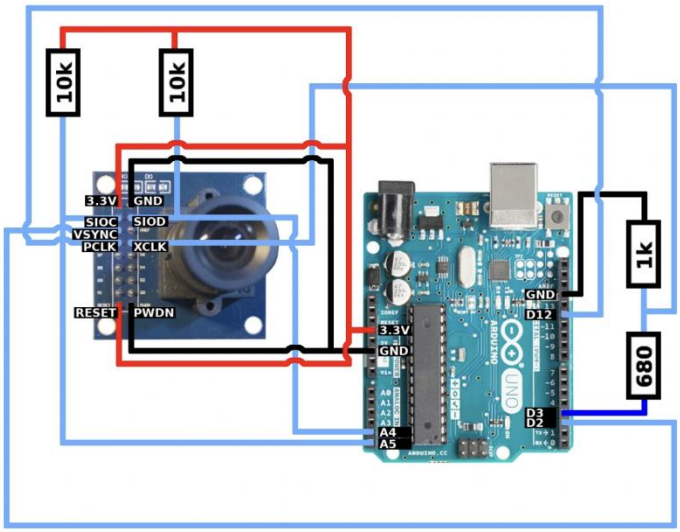


Fig 4: OV7670 - Arduino Connection

4.1.3 Arduino UNO R3

Connections from the OV7670 to the Arduino are made using UART Protocol. Leveraging the Arduino IDE via a laptop, we embed code into the microcontroller to pull images from the OV7670 camera to the Arduino.
Then we use software libraries via the IDE to pull the code into a PC/Local Server to consequently upload it to the internet.

The following software libraries are used:

- *Adafruit_ST7735_mod.cpp*
- *Adafruit_ST7735_mod.h*
- *ExampleGrayscale20HzInterlaced.cpp*
- *ExampleTftBufferedCameraFrame.cpp*
- *ExampleTftPixelByPixelCameraFrame.cpp*
- *ExampleUart.cpp*
- *GrayScaleTable.h*
- *LiveOV7670.ino*
- *setup.h*

Source:
https://github.com/indrekluuk/LiveOV7670/tree/master/src/LiveOV7670 and
https://circuitjournal.com/arduino-OV7670-to-pc

After installing these libraries, we need to wire the camera as in the image below. Using this wiring setup and the software libraries in place, we can obtain images, which can then be saved as a file on the laptop and sent to telegram via a workflow automation system.

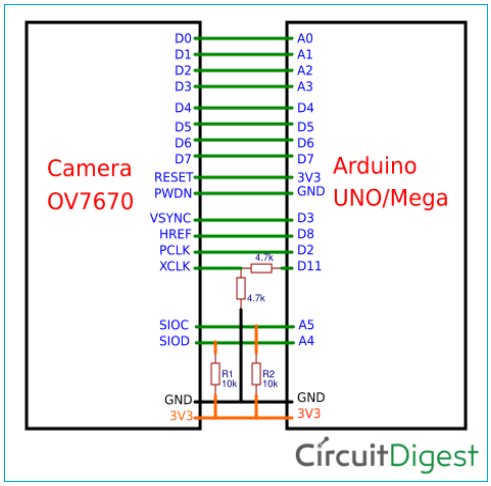


Fig 5: Arduino – OV7670 Wiring

4.1.4 Automation System

We have devised a Workflow Automation System to take the image from the Arduino serial monitor and instantaneously make it reach a telegram bot. Telegram is the platform through which a user can interact with DoorGuardian. It is a messaging app available on IOS, Android, Windows, MacOS and Linux operating systems as well as through a web browser, giving the user flexibility as well as the strength of a dynamic platform to interact with DoorGuardian.

Once the Ultrasonic System activates the power in the circuit, the OV7670 captures an image and sends it to the Arduino. Then, the captured images reach a PC Server/ Broadband Host via a wired connection.

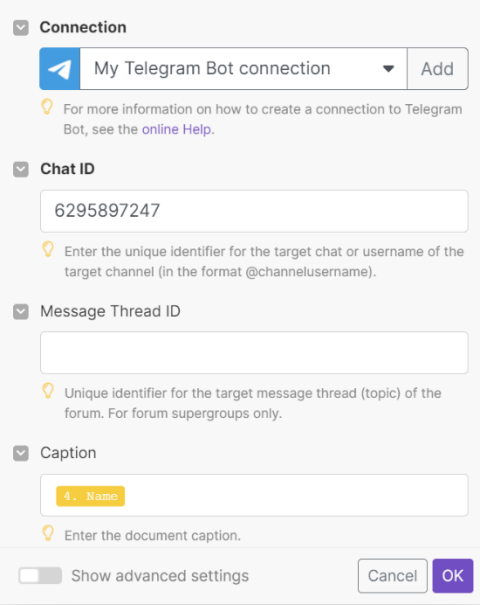


Fig 6: Telegram Workflow Automation

Then leveraging Cloud Infrastructure services such as Microsoft OneDrive, the image is securely uploaded to the internet. After the picture gets its own weblink, the images are put through a Workflow Automation System. We have used a third-party vendor called Make, which has modules which are the main building blocks of automation. Modules represent actions that Make performs which was a three-step process (i) watch the constantly updating Cloud Drive folder (ii) download new files as soon as they reach the folder (iii) send the files via a Telegram API, though a 'chat ID' obtained when we build our bot (see section 4.2.1) . Mapping links the 3, connecting the data retrieved by one module to another module to perform the desired action. This creates a 'scenario', i.e., the personalized workflow and it is made up of a series of modules that automate apps and services.

Part 2: Processing the Images and Operating the Door

4.2.1 The Telegram Bot

The telegram bot, which we developed ourselves with Telegram's API on the Arduino IDE, acts as the interface between the Arduino microcontroller and the motor based on user input.

In order to build the Telegram Bot, we used the Telegram Botfather to generate a new bot. This created a bot token which can be used in our code.

We also used the telegram IDBot to obtain the user ID and password, which is needed to ensure that only the desired user is able to use the bot to open or close their door.

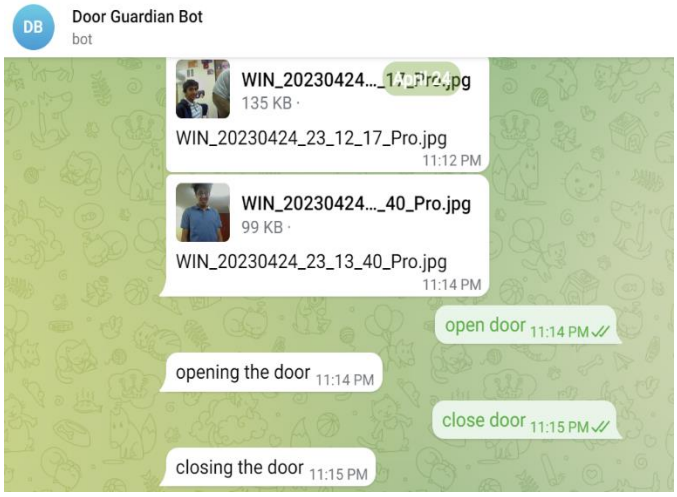


Fig 7: Telegram Bot Demonstration

Before beginning to use the Arduino IDE with the telegram API and this information, we first needed to install the following libraries:

1. Arduino Uno Wi-Fi Dev Ed Library
2. ArduinoJson
3. UniversalTelegramBot

Here are how the API calls are made so that the ESP32 Wi-Fi-Module can interact with Telegram Bot API:

Insert your network credentials in the designated variables.

1. Insider Network Credentialize to initialize the ESP module.
2. Set the GPIO you want to control (in this case, GPIO 2 is set to LOW by default).
3. Then we insert our Telegram Bot Token (obtained from Botfather).
4. Insert your Telegram User ID. Create a Wi-Fi client and a bot with the defined token and client.
5. Use the handleNewMessages() function to handle incoming messages. It stores the chat ID of the sender, saves the message content, and checks for specific commands.
6. To send a message, use the sendMessage() method on the bot object with the recipient's chat ID and the message.

7. Use the /start command to retrieve a list of valid commands.
8. Use the /motor_on command to turn on the Servo and update the ledState variable.
9. Use the /motor_off command to turn off the Sero and update the ledState variable.

The Telegram bot API provides a set of HTTP-based methods for sending and receiving messages, handling user input, and managing the bot's settings. The ESP32/ESP8266 microcontroller communicates with the Telegram bot through the use of HTTPS requests and responses, which are sent over the internet using the Wi-Fi client API. The handleNewMessages() function then processes the message and sends the appropriate commands to the microcontroller through the Wi-Fi client API. The bot also uses the sendMessage() method to send responses back to the user through the Telegram API.

The details of using these libraries with the telegram API, Arduino IDE and the ESP32 Module hardware is detailed in the next section.

4.2.2 ESP32 Wi-Fi Module

The ESP32 module is a microcontroller that acts as a Wi-Fi module in order to interconnect the SG90 servo motors and the telegram bot that we developed. The ESP32 module also uses the UART protocol in order to interact with devices on its system, however, we are not utilizing this feature of the module.

The ESP32 module links to your telegram ID and is connected to your phone over Wi-Fi. By entering in your Wi-Fi name and password, you can effectively establish a connection between the module and your smartphone and thus, by extension, you can control the motor using your phone.

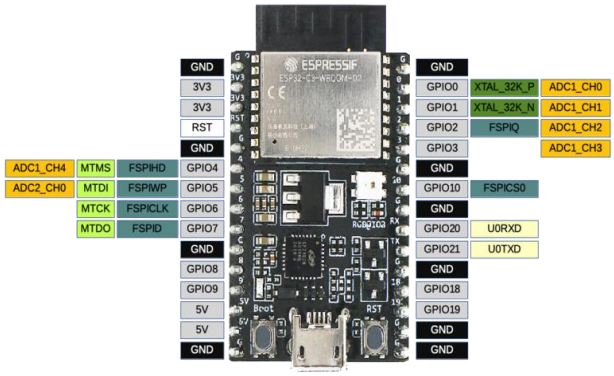


Fig 8: ESP32 Pin Layout

The code that we have used in order to establish connection between the 3 devices is given below:

```
#ifdef ESP32
#include <WiFi.h>
#else
#include <ESP8266WiFi.h>
#endif
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>   // Universal Telegram Bot Library w
https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
#include <ArduinoJson.h>

// Replace with your network credentials
const char* ssid = "E";
const char* password = "VayunGupta";

// Initialize Telegram BOT
#define BOTtoken "5945205082:AAHtkYmmtUdcnid8Y1DZeL7HhUWal
from Botfather)

// Use @myidbot to find out the chat ID of an individual or a group
// Also note that you need to click "start" on a bot before it can
// message you
#define CHAT_ID "6295897247"

#ifdef ESP8266
X509List cert(TELEGRAM_CERTIFICATE_ROOT);
#endif

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

// Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

const int motorpin = 2;
bool motorstate = LOW;

// Handle what happens when you receive new messages
void handleNewMessages(int numNewMessages) {
  Serial.println("handleNewMessages");
  Serial.println(String(numNewMessages));

  for (int i=0; i<numNewMessages; i++) {
    // Chat id of the requester
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID){
      bot.sendMessage(chat_id, "Unauthorized user", "");
```

```
      continue;
    }

    // Print the received message
    String text = bot.messages[i].text;
    Serial.println(text);

    String from_name = bot.messages[i].from_name;

    if (text == "/start") {
      String welcome = "Welcome, " + from_name + ".\n";
      welcome += "Use the following commands to control your outputs.\n\n";
      welcome += "/led_on to turn GPIO ON \n";
      welcome += "/led_off to turn GPIO OFF \n";
      welcome += "/state to request current GPIO state \n";
      bot.sendMessage(chat_id, welcome, "");
    }

    if (text == "door open") {
      bot.sendMessage(chat_id, "opening the door", "");
      motorstate = HIGH;
      digitalWrite(motorpin, motorstate);
    }

    if (text == "door close") {
      bot.sendMessage(chat_id, "closing the door", "");
      motorstate = LOW;
      digitalWrite(motorpin, motorstate);
    }

    if (text == "/state") {
      if (digitalRead(motorpin)){
        bot.sendMessage(chat_id, "door open", "");
      }
      else{
        bot.sendMessage(chat_id, "door close", "");
      }
    }
  }
}

void setup() {
  Serial.begin(115200);

  #ifdef ESP8266
    configTime(0, 0, "pool.ntp.org");
    client.setTrustAnchors(&cert);
  #endif

  pinMode(motorpin, OUTPUT);
  digitalWrite(motorpin, motorstate);

  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  #ifdef ESP32
  client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
  #endif
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  // Print ESP32 Local IP Address
  Serial.println(WiFi.localIP());
}

void loop() {
  if (millis() > lastTimeBotRan + botRequestDelay)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while(numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
}
```

4.2.3 Motor and Motor Driver

For the purpose of opening and closing the lock, we are using SG90 servo motors. The servo motors have the unique ability of moving in both clockwise and anti-clockwise directions, which is very helpful in the application of our project since this portrays the locking and unlocking of our door.

The servo motors have in-built motor drivers which make our process of integration much simpler since we do not have to establish a buffer connection between the Arduino and the motor. The servo motor has only 3 wires, one that establishes connection to power, one that goes to ground, and the other that is used as the signal pin. We have connected this module to the ESP32 controller and thus, the owner of the house can send a signal to his telegram bot in order to control the functioning of the motor, and by extension, he can open and close the lock.

Fig 9: The Servo Motor

The servo motor has a duty cycle of 1-2 milliseconds and the PWM period for the same, that is, the period throughout transmission of data is 20 milliseconds. It operates at a voltage of 5V. It is connected through pin 12 of the ESP32.

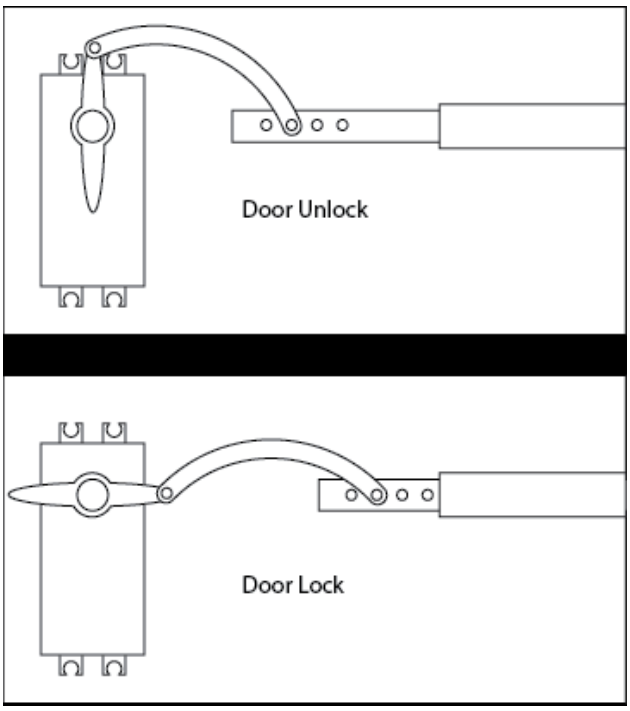


Fig 10: Motorized Lock

5. Results and Conclusions

Putting all the different components together was a challenging procedure, as it required the integration of UART, WIFI, wiring and other communication protocols. The final result, however, was worth it, as we learned about a wide range of fields such as:

1. Automation
2. Cyber-security
3. Internet of Things
4. Networking
5. APIs
6. App development

We plan to continue working on and increasing the scope of our project over the summer, adding the need for Artificial Intelligence, Web development, and Backend development.

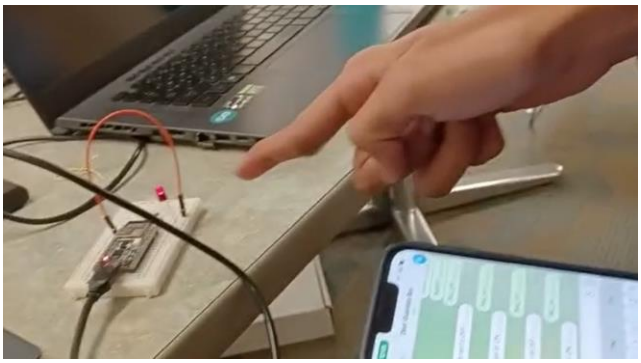


Fig 11: The first successful integration of the Telegram Bot with the ESP32 was one of the highlights of our building process.

6. References:

API:
https://aws.amazon.com/what-is/api/

UART Protocol:
https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

Ultrasonic:
https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

OV7670 Camera Module:
https://www.youtube.com/watch?v=R94WZH8XAvM

Automation System:
https://www.make.com/en/integrations/telegram

Telegram Bot and ESP32 Module:
Lesson24- how to use Telegram control KC868-A4 A8 (ESP32) relay output (kincony.com)

Telegram: Control ESP32/ESP8266 Outputs with Arduino IDE | Random Nerd Tutorials

https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot

Motor and Motor Driver:
https://circuitdigest.com/microcontroller-projects/interfacing-sg90-servo-motor-with-esp32

https://dronebotworkshop.com/esp32-servo/