DOOR LOCK SYSTEM USING ESP32 CAM

A PROJECT BASED LEARNING REPORT

Submitted by

PRADEEP .C	**(21EC057)**
RANJITH .M	**(21EC064)**
NITHISH KUMAR . C	**(21EC055)**
VISHNUPRIYA .Y	**(21EC103)**
VIKRAM .P	**(21ECL26)**

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

NANDHA ENGINEERING COLLEGE, ERODE

(AUTONOMOUS)

(Affiliated to Anna University, Chennai)

MAY 2023

NANDHA ENGINEERING COLLEGE, ERODE

(AUTONOMOUS)

(Affiliated to Anna University, Chennai)

Certified that this project report "**DOOR LOCK SYSTEM USING ESP32 CAM"** is the bonafide work of **"VISHNUPRIYA.Y, NITHISHKUMAR . C, PRADEEP.C, RANJITH.M ,VIKRAM.P"who** carried out the Project Based Learning work under my supervision.

SIGNATURE,

Mr. E K. ARULKARTHIK, M.E

Assistant Professor/ECE

PBL COORDINATOR,

Department of ECE,

Nandha Engineering College,

(Autonomous)

Erode-638052.

ACKNOWLEDGEMENT

The success of a work depends on the team work and co-operation various involved either directly or indirectly.

We express our thanks to our beloved Chairman of Nandha Institutions **Thiru.V.SHANMUGAN** for providing us all the basic amenities to complete the course successfully.

We wish to convey our earnest gratefulness to our cherished Secretary **Thiru.S.THIRUMOORTHI** for the excellent facilities provided to the project work successfully.

We wish to express our deep sense of gratitude and thanks to our beloved Principal **Dr.N.RENGARAJAN** for the encouragement and support during the course study.

We are grateful to our DEAN **Dr.S.KAVITHA** and our Project Based Learning Coordinator Mr. **G.PRABHAKARAN** Electronics and Communication Engineering for providing the facilities and also for their kind patronage.

We thank all the staff members of Electronics and Communication Engineering Department for their valuable suggestions in the project.

TITLE OF CONTENT

ABSTRACT:

In this ESP32 CAM project, I have explained how to make WiFi door lock with photo capture using ESP32-CAM and Telegram app. With this IoT project you can take multiple photo, unlock and lock the door from anywhere in the world with the Telegram app. When anyone presses the doorbell, you will get a notification **in the telegram app with a** photo **of that person. After that, you can easily** unlock **and** lock **the door from the telegram app. So this IoT project also can be used as a home security system. In this article, I have explained all the steps to make this WiFi door lock with photo capture.**

CHAPTER 1

INTRODUCTION

Smart doors are widely used in many applications. Now days everyone has smartphones and having internet access easily. IOT **based systems and accessing units are very helpful in everywhere. Here we want to develop a smart door control with telegram app. We can access telegram app from pc or mobile phone. We can lock and unlock door by controlling valve from telegram app. Also it will send photo whenever doing lock and unlock. We can access this system from anywhere. Here the project title is Smart door lock from telegram app with *ESP32-CAM*.**

1.1 Hardware required:

The circuit diagram of the door lock system using ESP32 and telegram is shown in the above figure. The hardware requirements are

- ESP32-CAM board
- 12V Electronic Lock
- TIP122 NPN Transistor
- 7805 5V regulator
- 1N4007 diode
- 1k 0.25-watt Resistor
- 10k 0.25-watt Resistor
- 100uF 25V DC Capacitor
- Push Switch
- 12V DC supply

CHAPTER 2

2.1 Introduction to ESP32 CAM

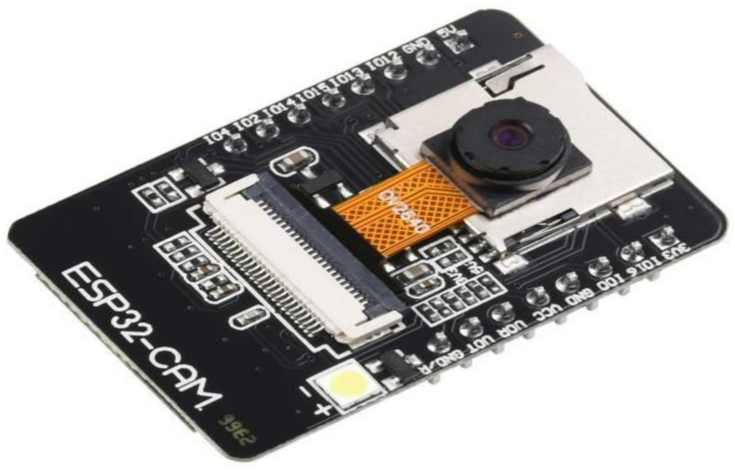

IO4 IO2 IO14 IO15 IO13 IO12 GND 5V
OV2640
ESP32-CAM
3V3 IO16 IO0 GND VCC U0R U0T GND/R

Figure 2.1. ESP32 CAM

The ESP32-CAM is a small size, low power consumption camera module based on ESP32. It comes with an OV2640 camera and an ESP32-CAM-MB micro usb to serial port adapter.

The ESP32-CAM can be widely used in intelligent IoT applications such as wireless video monitoring, WiFi image upload, QR identification, and so on.

Features

- Onboard ESP32-S module, supports WiFi + Bluetooth
- OV2640 camera with flash
- Onboard TF card slot, supports up to 4G TF card for data storage
- Supports WiFi video monitoring and WiFi image upload

- Supports multi sleep modes, deep sleep current as low as 6mA
- Control interface is accessible via pinheader, easy to be integrated and embedded into user products
- Comes with ESP32-CAM-MB Micro USB to serial port adapter, adapts CH340 chip, for connecting ESP32-CAN to the PC, no additional converter required

2.2 What's on board

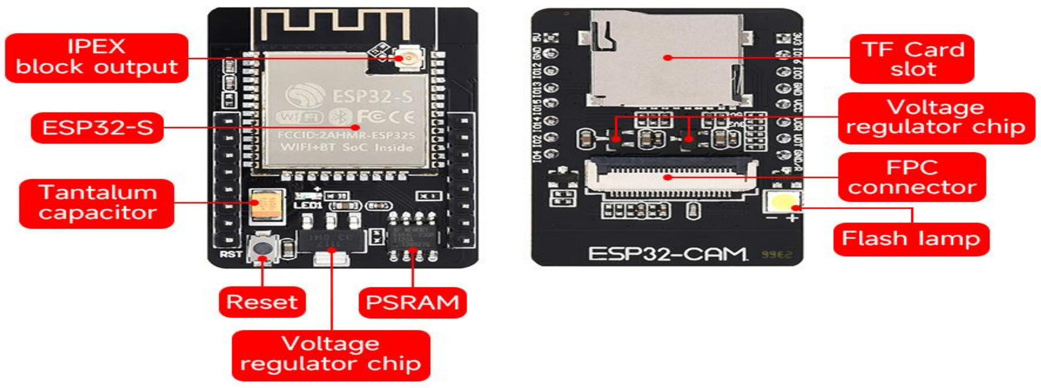


Figure 2.2. whats on Board

CHAPTER 3

3.1 Introduction to SOLENOID LOCK

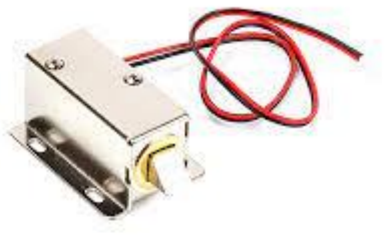


Figure 3.1 SOLENOID LOCK

An **electronic lock** (or **electric lock**) is a locking device which operates by means of electric current. Electric locks are sometimes stand-alone with an electronic control assembly mounted directly to the lock. Electric locks may be connected to an access control system, the advantages of which include: key control, where keys can be added and removed without re-keying the lock cylinder; fine access control, where time and place are factors; and transaction logging, where activity is recorded. Electronic locks can also be remotely monitored and controlled, both to lock and to unlock.

3.2 Operation

Electric locks use magnets, solenoids, or motors to actuate the lock by either supplying or removing power. Operating the lock can be as simple as using a switch, for example an apartment intercom door release, or as complex as a biometric based access control system.

There are two basic types of locks: "preventing mechanism" or operation mechanism.

CHAPTER 4

4.1 Introduction to TIP 122 Transistor

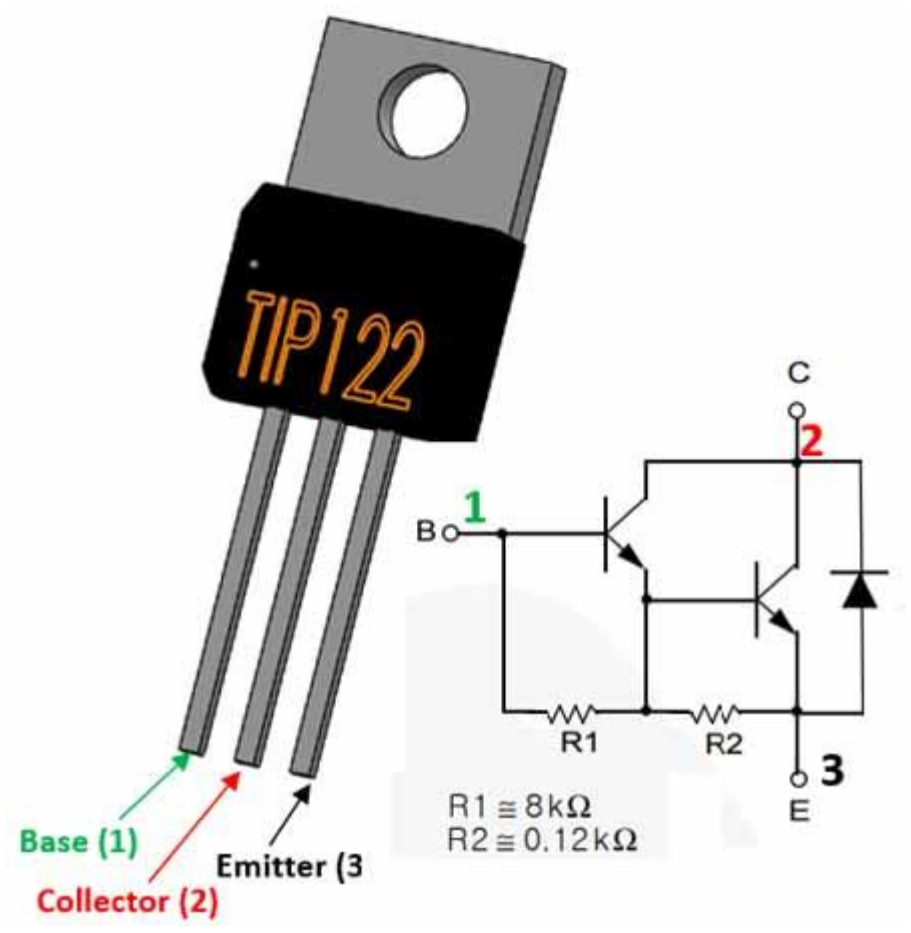
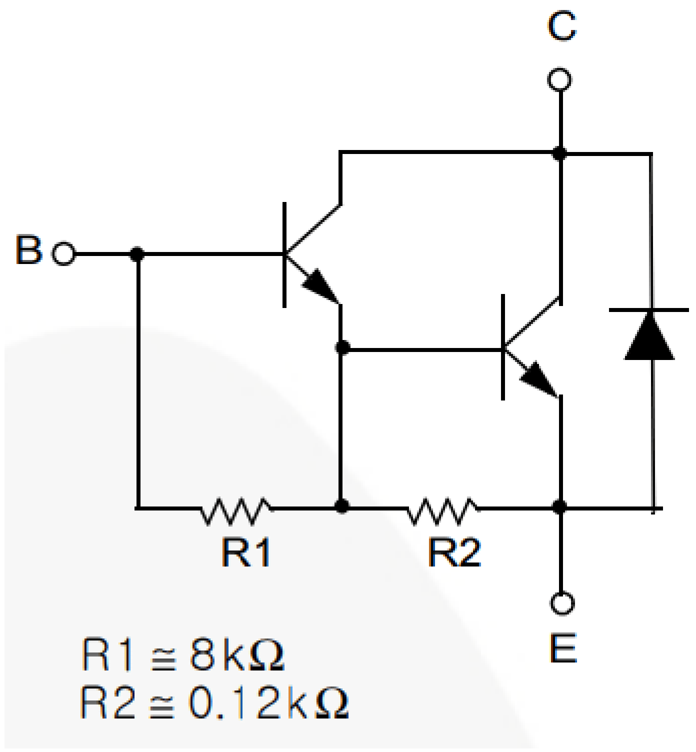


Figure 4.1. tip 122 transistor

The **TIP122** is a Darlington pair NPN transistor. It functions like a normal NPN transistor, but since it has a Darlington pair inside it has a good collector current rating of about 5A and a gain of about 1000. It can also withstand about 100V across its collector- Emitter hence can be used to drive heavy loads. The Darlington pair inside this transistor is shown clearly as its internal circuit schematic below



As you can see, there are two transistors inside this TO-220 package in which the emitter of the first transistor is connected with the base of the second transistor and the collector of both transistors are connected together to form a Darlington pair. This increases the current gain and current rating of this transistor.

4.2 7805 5v REGULATOR

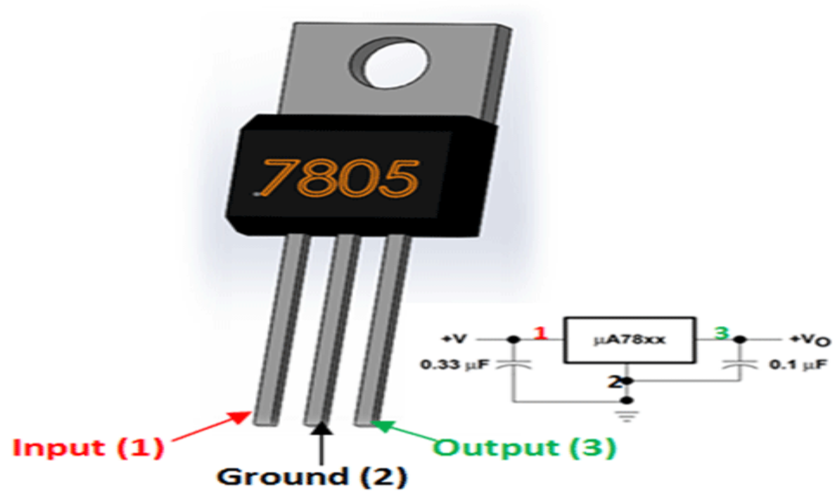


Figure 4.2. 7805 5v REGULATOR

The **7805 Voltage Regulator IC** is a commonly used voltage regulator that finds its application in most of the electronics projects. It provides a constant +5V output voltage for a variable input voltage supply.

LM7805 Pinout Configuration

Pin Number	**Pin Name**	**Description**
1	Input (V+)	Unregulated Input Voltage
2	Ground (Gnd)	Connected to Ground
3	Output (Vo)	Outputs Regulated +5V

7805 Regulator Features

- 5V Positive Voltage Regulator
- Minimum Input Voltage is 7V
- Maximum Input Voltage is 25V
- Operating current(I_Q) is 5mA
- Internal Thermal Overload and Short circuit current limiting protection is available.
- Junction Temperature maximum 125 degree Celsius
- Available in TO-220 and KTE package

4.3 DIODE

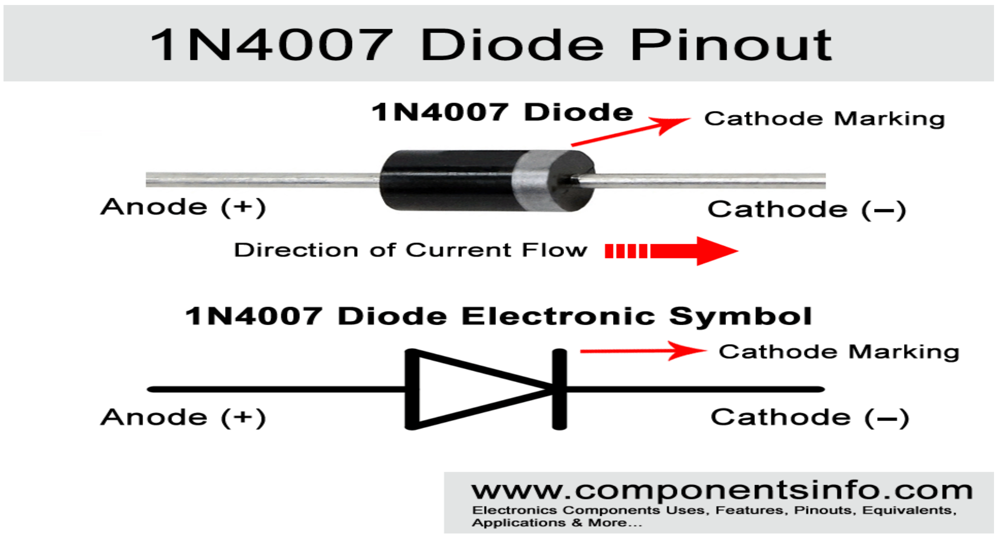

1N4007 Diode Pinout
1N4007 Diode
Cathode Marking
Anode (+)
Cathode (–)
Direction of Current Flow
1N4007 Diode Electronic Symbol
Cathode Marking
Anode (+)
Cathode (–)
www.componentsinfo.com
Electronics Components Uses, Features, Pinouts, Equivalents, Applications & More...

Figure 4.3 DIODE

A 1N4007 is a widely used general purpose diode. It is normally build to use as rectifier in the power supplies section of electronic appliances for converting AC voltage to DC with other filter capacitors. It is a diode of 1N400x series in which there are also other similar diodes from **1N4001** to 1N4007 and the only difference between them is the max repetitive reverse voltage.
Moreover it can also be used in any general purpose application where there is need of a general diode. The 1N4007 diode is built for working with high voltages and it can easily handle voltage below 1000V. The 1000mA or 1A average fwd current, 3W power dissipation with small size and lost cost also makes it ideal for wide variety of applications.

4.4 CAPACITOR

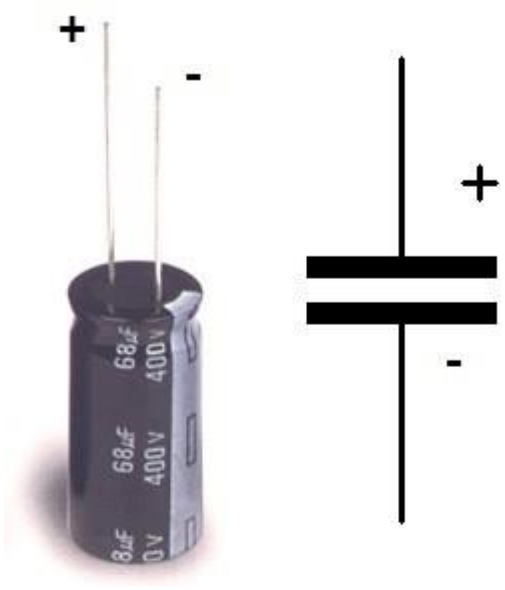


Fig.4.4. CAPACITOR

A **capacitor** is a device that stores electrical energy in an electric field by virtue of accumulating electric charges on two close surfaces insulated from each other. It is a passive electronic component with two terminals.

The effect of a capacitor is known as capacitance. While some capacitance exists between any two electrical conductors in proximity in a circuit, a capacitor is a component designed to add capacitance to a circuit. The capacitor was originally known as the **condenser**, a term still encountered in a few compound names, such as the *condenser microphone*.

The physical form and construction of practical capacitors vary widely and many types of capacitor are in common use. Most capacitors contain at least two electrical conductors often in the form of metallic plates or surfaces separated by a dielectric medium. A conductor may be a foil, thin film, sintered bead of metal, or an electrolyte. The nonconducting dielectric acts to increase the capacitor's charge capacity. Materials commonly used as dielectrics include glass, ceramic, plastic film, paper, mica, air, and oxide layers. Capacitors are widely used as parts of electrical circuits in many common electrical devices. Unlike a resistor, an ideal capacitor does not dissipate energy, although real-life capacitors do dissipate a small amount (see Non-ideal behavior). When an electric potential difference (a voltage) is applied across the terminals of a capacitor, for example when a capacitor is connected across a battery, an electric field develops across the dielectric, causing a net positive charge to collect on one plate and net negative charge to collect on the other plate. No current actually flows through the dielectric. However, there is a flow of charge through the source circuit. If the condition is maintained sufficiently long, the current through the source circuit ceases. If a time-varying voltage is applied across the leads of the capacitor, the source experiences an ongoing current due to the charging and discharging cycles of the capacitor.

4.7 PUSH SWITCH

Figure 4.7. PUSH SWITCH

A **push switch (button)** is a momentary or non-latching switch which causes a temporary change in the state of an electrical circuit only while the switch is physically actuated. An automatic mechanism (i.e. a spring) returns the switch to its default position immediately afterwards, restoring the initial circuit condition. There are two types:[1]

- A 'push to make' switch allows electricity to flow between its two contacts when held in. When the button is released, the circuit is broken. This type of switch is also known as a Normally Open (NO) Switch. (Examples: doorbell, computer case power switch, calculator buttons, individual keys on a keyboard)
- A 'push to break' switch does the opposite, i.e. when the button is not pressed, electricity can flow, but when it is pressed the circuit is broken. This type of switch is also known as a Normally Closed (NC) Switch. (Examples: Fridge Light Switch, Alarm Switches in Fail-Safe circuits)

Many Push switches are designed to function as both 'push to make' and 'push to break' switches. For these switches, the wiring of the switch determines whether the switch functions as a 'push to make' or as a 'push to break' switch.

CHAPTER 5
DOOR LOCK SYSTEM USING ESP32

5.1 Circuit diagram:

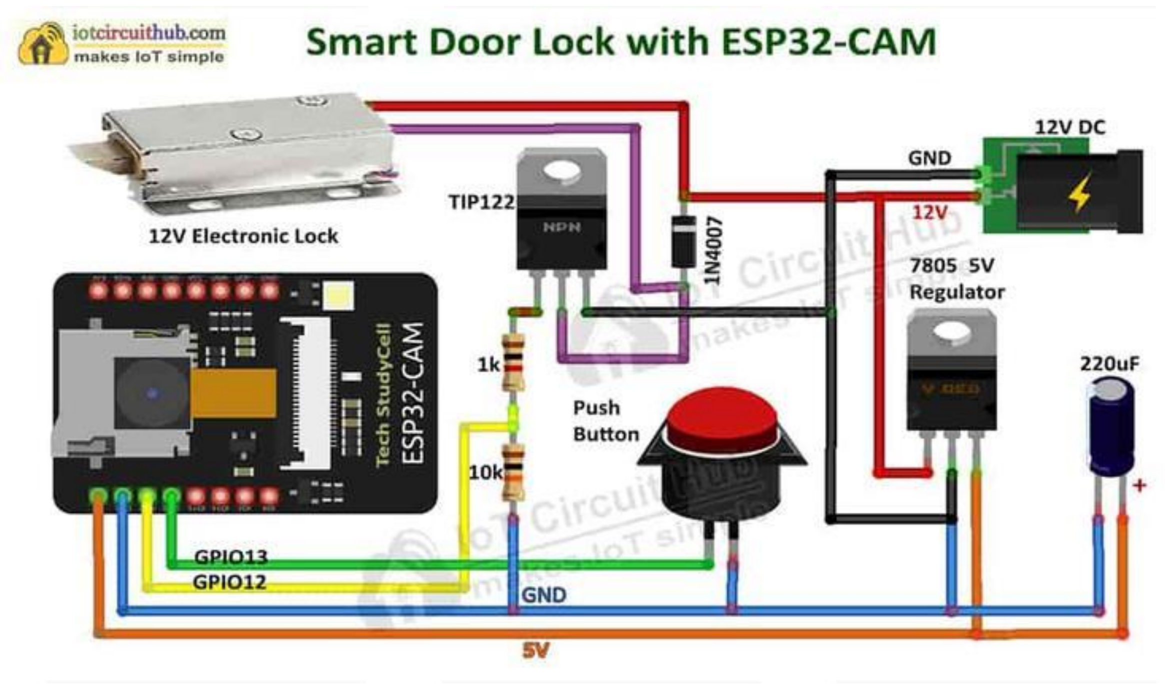


Figure 5.1. circuit diagram

5.3 Working:

ESP32-CAM placed on daughter board and it is having 5V power supply and all digital pin outs. We can easily mounted ESP32-CAM on this daughter board for easy connectivity. Relay Connected to ESP32-CAM digital pin and it can control solenoid valve. Also toggle button connected to ESP32-CAM digital pin. This digital pin helps us to send request notification to telegram app. After pressing this button we will get notification on telegram app along with photo. Then we can unlock door from app. Green LED will be ON while accessing ESP32-CAM with telegram app. This smart door lock is very effective and communication is very fast than any other cloud based IOT apps or web systems.

CHAPTER 6

6.1 APPLICATIONS of the project

- Smart Doors
- Remote control door access
- Garage controls
- Shutters control applications

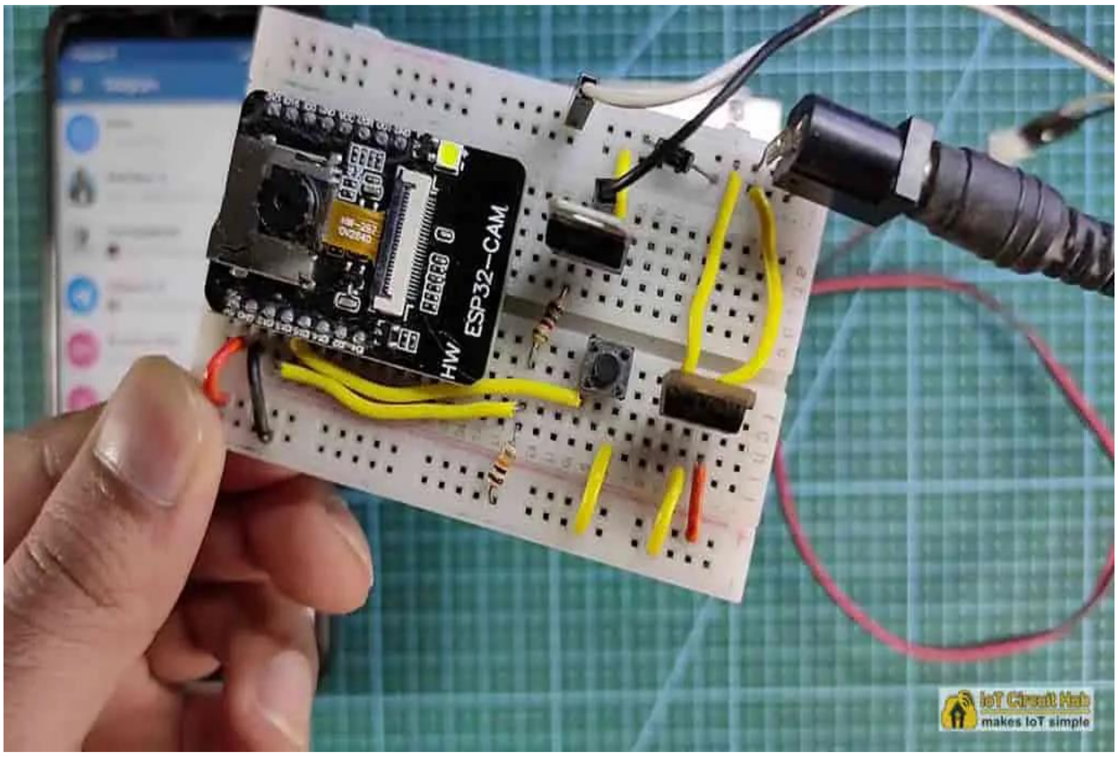
6.2 SOURCE CODE

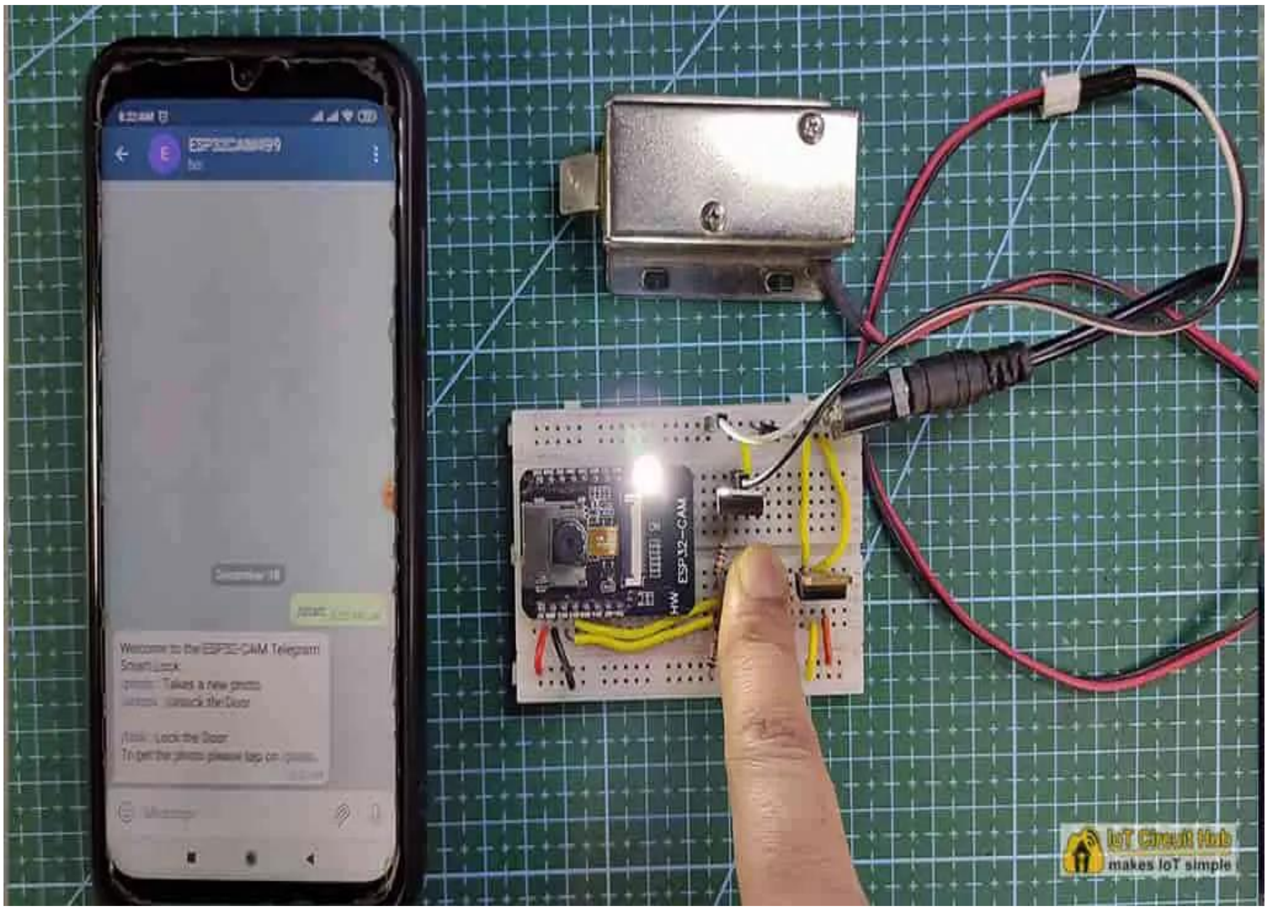
```
/**********************************************************************************
 *  TITLE: ESP32CAM Telegram WiFi Door Lock with photo capture
 *  Click on the following links to learn more.
 *  YouTube Video: https://youtu.be/11V2ZzHpW3Q
 *  Related Blog : https://iotcircuithub.com/esp32-cam-telegram-wifi-door-lock
 *  by Tech StudyCell
 *  Preferences--> Aditional boards Manager URLs :
 *  https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json
 *
 *  Download Board ESP32 : https://github.com/espressif/arduino-esp32
 *  Download the libraries
 *  Brian Lough's Universal Telegram Bot Library: https://github.com/witnessmenow/Universal-Arduino-
Telegram-Bot
 **********************************************************************************/


#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

// Replace with your network credentials
const char* ssid = "";  //WiFi Name
const char* password = "";  //WiFi Password

// Use @myidbot to find out the chat ID of an individual or a group
// You need to click "start" on a bot before it can message you
// Initialize Telegram BOT
String chatId = "XXXXXXXXXX";
String BOTtoken = "XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

bool sendPhoto = false;
```

```
WiFiClientSecure clientTCP;

UniversalTelegramBot bot(BOTtoken, clientTCP);

// Define GPIOs
#define BUTTON 13
#define LOCK 12
#define FLASH_LED 4

//CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22


int lockState = 0;
String r_msg = "";

const unsigned long BOT_MTBS = 1000; // mean time between scan messages
unsigned long bot_lasttime; // last time messages' scan has been done

void handleNewMessages(int numNewMessages);
String sendPhotoTelegram();

String unlockDoor(){
 if (lockState == 0) {
  digitalWrite(LOCK, HIGH);
  lockState = 1;
  delay(100);
  return "Door Unlocked. /lock";
 }
 else{
  return "Door Already Unlocked. /lock";
 }
}
String lockDoor(){
 if (lockState == 1) {
  digitalWrite(LOCK, LOW);
  lockState = 0;
  delay(100);
  return "Door Locked. /unlock";
 }
 else{
  return "Door Already Locked. /unlock";
 }
```

```
}

String sendPhotoTelegram(){
  const char* myDomain = "api.telegram.org";
  String getAll = "";
  String getBody = "";

  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    delay(1000);
    ESP.restart();
    return "Camera capture failed";
  }

  Serial.println("Connect to " + String(myDomain));

  if (clientTCP.connect(myDomain, 443)) {
    Serial.println("Connection successful");

   Serial.println("Connected to " + String(myDomain));

    String head = "--IotCircuitHub\r\nContent-Disposition: form-data; name=\"chat_id\"; \r\n\r\n" + chatId +
"\r\n--IotCircuitHub\r\nContent-Disposition: form-data; name=\"photo\"; filename=\"esp32-
cam.jpg\"\r\nContent-Type: image/jpeg\r\n\r\n";
    String tail = "\r\n--IotCircuitHub--\r\n";

    uint16_t imageLen = fb->len;
    uint16_t extraLen = head.length() + tail.length();
    uint16_t totalLen = imageLen + extraLen;

    clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto HTTP/1.1");
    clientTCP.println("Host: " + String(myDomain));
    clientTCP.println("Content-Length: " + String(totalLen));
    clientTCP.println("Content-Type: multipart/form-data; boundary=IotCircuitHub");
    clientTCP.println();
    clientTCP.print(head);

    uint8_t *fbBuf = fb->buf;
    size_t fbLen = fb->len;
    for (size_t n=0;n<fbLen;n=n+1024) {
      if (n+1024<fbLen) {
        clientTCP.write(fbBuf, 1024);
        fbBuf += 1024;
      }
      else if (fbLen%1024>0) {
        size_t remainder = fbLen%1024;
        clientTCP.write(fbBuf, remainder);
      }
    }

    clientTCP.print(tail);

    esp_camera_fb_return(fb);

    int waitTime = 10000;   // timeout 10 seconds
    long startTimer = millis();
    boolean state = false;
    while ((startTimer + waitTime) > millis()){
      Serial.print(".");
      delay(100);
      while (clientTCP.available()){
          char c = clientTCP.read();
          if (c == '\n'){
            if (getAll.length()==0) state=true;
            getAll = "";
          }
          else if (c != '\r'){
            getAll += String(c);
          }
          if (state==true){
            getBody += String(c);
          }
          startTimer = millis();
       }
       if (getBody.length()>0) break;
    }
    clientTCP.stop();
    Serial.println(getBody);
  }
  else {
    getBody="Connected to api.telegram.org failed.";
    Serial.println("Connected to api.telegram.org failed.");
  }
  return getBody;
}

void handleNewMessages(int numNewMessages){
  Serial.print("Handle New Messages: ");
  Serial.println(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 != chatId){
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
    }

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

    String fromName = bot.messages[i].from_name;
    if (text == "/photo") {
      sendPhoto = true;
      Serial.println("New photo request");
    }
    if (text == "/lock"){
      String r_msg = lockDoor();
      bot.sendMessage(chatId, r_msg, "");
    }
    if (text == "/unlock"){
      String r_msg = unlockDoor();
      bot.sendMessage(chatId, r_msg, "");
    }
    if (text == "/start"){
      String welcome = "Welcome to the ESP32-CAM Telegram Smart Lock.\n";
      welcome += "/photo : Takes a new photo\n";
      welcome += "/unlock : Unlock the Door\n\n";
      welcome += "/lock : Lock the Door\n";
      welcome += "To get the photo please tap on /photo.\n";
      bot.sendMessage(chatId, welcome, "Markdown");
    }
  }
}

void setup(){
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  Serial.begin(115200);
  delay(1000);

  pinMode(LOCK,OUTPUT);
  pinMode(FLASH_LED,OUTPUT);
  pinMode(BUTTON,INPUT_PULLUP);

  digitalWrite(LOCK, LOW);

  WiFi.mode(WIFI_STA);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("ESP32-CAM IP Address: ");
  Serial.println(WiFi.localIP());

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;  //0-63 lower number means higher quality
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;  //0-63 lower number means higher quality
    config.fb_count = 1;
  }

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    delay(1000);
    ESP.restart();
  }

  // Drop down frame size for higher initial frame rate
  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s, FRAMESIZE_CIF);  // UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
}

void loop(){

  if (sendPhoto){
    Serial.println("Preparing photo");
    digitalWrite(FLASH_LED, HIGH);
    delay(200);
    sendPhotoTelegram();
    digitalWrite(FLASH_LED, LOW);
    sendPhoto = false;
  }


  if(digitalRead(BUTTON) == LOW){
    Serial.println("Preparing photo");
    digitalWrite(FLASH_LED, HIGH);
    delay(200);
    sendPhotoTelegram();
    digitalWrite(FLASH_LED, LOW);
    sendPhoto = false;
  }

  if (millis() - bot_lasttime > BOT_MTBS)
  {
    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);
    }
    bot_lasttime = millis();
  }
}
```

6.3 Working prototype :





6.4 Conclution

This is a very useful project for home security systems. As you can control the door lock and capture the photo from anywhere in the world through the internet.

6.5 Reference:

https://iotcircuithub.com/esp32-cam-telegram-wifi-door-lock/