VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

DESIGNING WIRELESS-EMBEDDED SYSTEM

FINAL REPORT

REMOTE-CONTROLLED CAMERA ROBOT USING IOT

Instructor: PhD Đoàn Duy

Student Group:

Nguyễn Thành Tài	21522568
Nguyễn Đình Sơn	21522254

CLASS: CE232.O21. MTCL

HO CHI MINH CITY, May 2024

Tables of Contents:

I. Introduction

- In the age of ever-evolving technology, the convergence of robotics and Internet of Things (IoT) has paved the way for innovative solutions in various domains. One such groundbreaking endeavor is the development of a remote-controlled camera robot, leveraging the power of IoT to revolutionize surveillance, exploration, and remote monitoring applications.

A Surveillance Robot

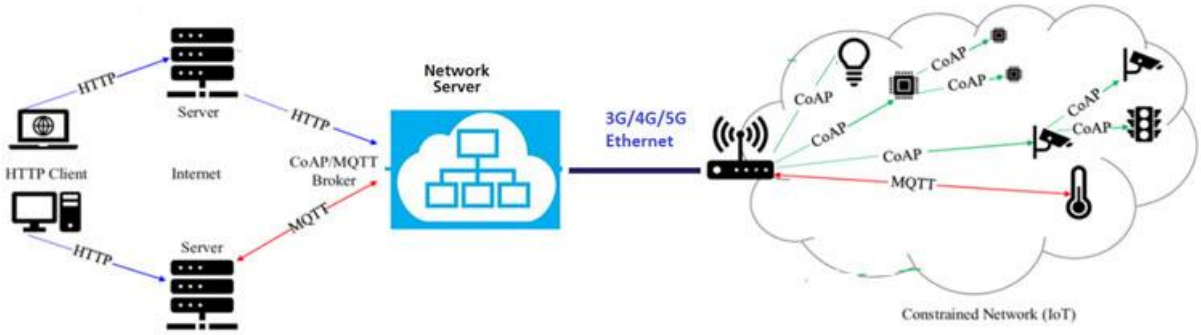
- The “Remote-Controlled Camera Robot Using IoT” project aims to address the need for flexible and efficient surveillance systems that can adapt to diverse environments and scenarios, empowering users with remote control and real-time monitoring capabilities.

- By integrating IoT principles into the design and functionality of the camera robot, this project offers unparalleled convenience and accessibility. Users can remotely manipulate the robot's movement

and camera orientation through a user-friendly interface, accessible via smartphones, tablets, or computers.

- Key features of the Remote-Controlled Robot Using IoT project include:

    - Wireless Connectivity: Leveraging IoT protocols for wireless communication, enabling remote control and data transmission over the internet.
    - Pan-and-Tilt Camera Mechanism: Equipped with a versatile camera system capable of pan-and-tilt movements for comprehensive coverage and surveillance.
    - Real-Time Monitoring: Providing users with live video feeds and sensor data, ensuring real-time monitoring and analysis of remote environments.



IoT Wireless Network Protocols

⇨ In summary, the "Remote-Controlled Camera Robot Using IoT" project represents a significant advancement in the realm of remote surveillance and exploration. By harnessing the synergy between robotics and IoT, this project sets a new standard for efficiency, accessibility, and functionality in remote-controlled camera systems, with promising implications across various industries and applications.

II. Hardware System

1. NodeMCU ESP32 Module

- NodeMCU ESP32 Kit is a Wi-Fi and Bluetooth transceiver KIT based on the ESP32 Wi-Fi SoC chip and the powerful CP2102 communication chip. Used for applications that need to connect, collect data and control via Wi-Fi, via Bluetooth, especially applications related to IoT. Using the latest ESP32-WROOM-32 Module with Dual core. With the easy-to-use design of the Arduino IDE's direct compiler for browser installation and code downloading, this makes using and installing applications via Wi-Fi and Bluetooth on the ESP32 very simple.

NodeMCU ESP32 Module

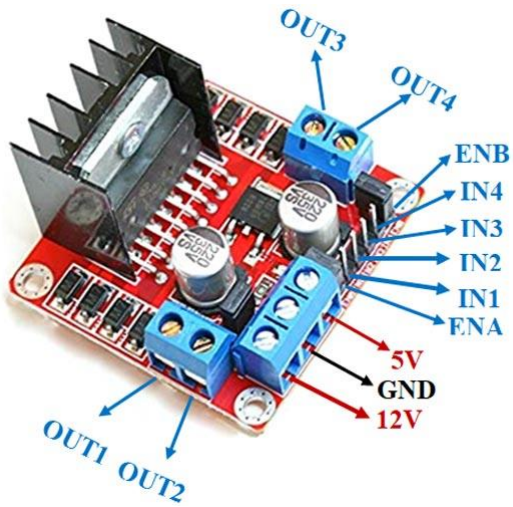
2. ESP32-CAM AI THINKER

- ESP32-CAM AI-Thinker is developed on the central microcontroller platform ESP32 SoC with the latest Wi-Fi, BLE technology and ARM architecture today, the kit combined with the OV2640 Camera uses in image transmission applications, image processing via Wi-Fi, Bluetooth or IoT applications, the circuit has good machining quality and high durability. In this project, we will use this ESP32-CAM to stream real time video via Wi-Fi.

ESP32-CAM AI THINKER

3. L298N Motor Driver Module

- L298 DC Motor Driver motor control circuit can control 2 DC motors, maximum current of 2A per motor, integrated protection diode circuit and 7805 power IC to help supply 5VDC power to other modules.



L298N Motor Driver Module

L298N Module Pinout Configuration

Pin Name	Description
IN1 & IN2	Motor A input pins. Used to control the spinning direction of Motor A
IN3 & IN4	Motor B input pins. Used to control the spinning direction of Motor B
ENA	Enables PWM signal for Motor A
ENB	Enables PWM signal for Motor B
OUT1 & OUT2	Output pins of Motor A
OUT3 & OUT4	Output pins of Motor B
12V	12V input from DC power Source
5V	Supplies power for the switching logic circuitry inside L298N IC
GND	Ground pin

4. DC Geared Motor

- DC Gear motor is also called DC Geared Motor. It consists of an electric DC motor and a gearbox or gearhead; these gearheads are used to reduce the DC motor speed, while increasing the DC motor torque.

DC Geared motor

- Operating voltage: 3V – 12V
- Maximum Torque: 800g/cm at 3V

5. LED

- We will use an additional LED in the robot to be able to use it for lighting in the dark.

LED 5mm

- Operating Voltage: 2.8 – 3.2 V
- Current: 5mA – 40mA

6. Servo SG90 Motor

- The servo mounted on the front of the robot helping the robot's camera rotate to different angles so it can see in different directions.

SG90 Servo Motor

- Pin configuration for the servo:
  - Red wire (VCC): Connect to a 5V pin.
  - Black wire (GND): Connect to a GND pin on the ESP32.
  - Yellow wire (signal): Connect to 3.3V signal or 5V signal.

7. DC LM2596 Module

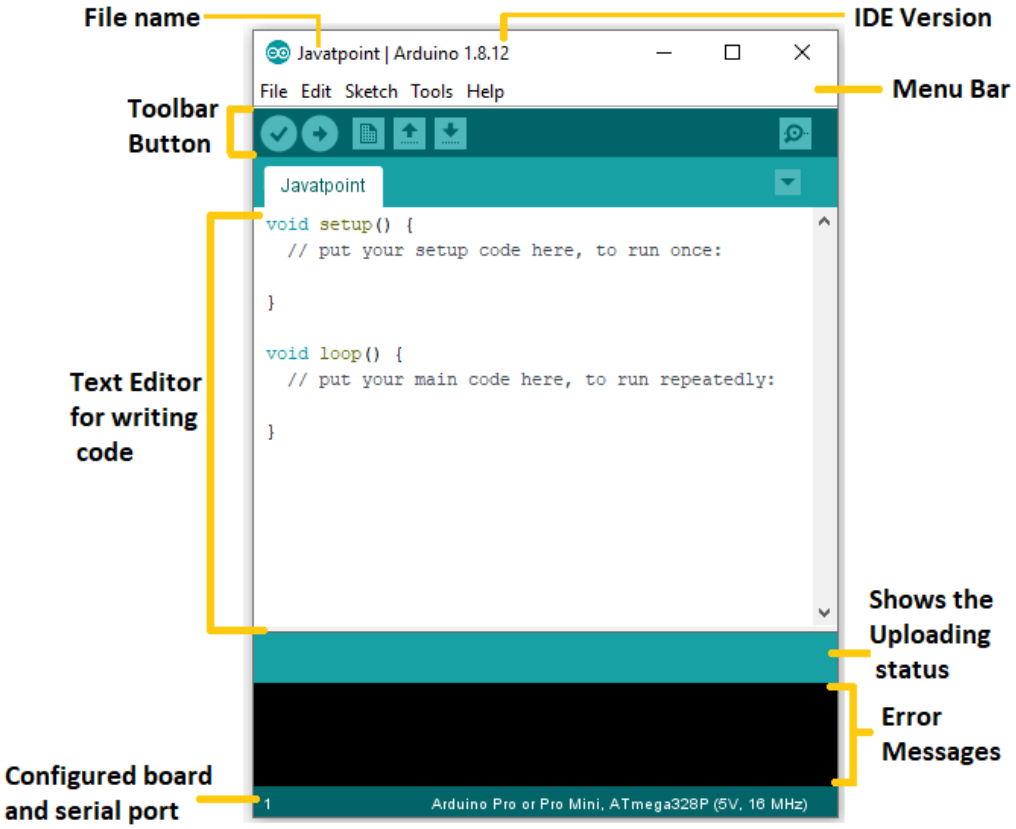
- LM2596 module is used to control the servo motor (operates with 5V voltage)

- Input voltage: 3V – 30V.
- Output voltage: 1.5V – 30V.
- Maximum current: 3A
- Power: 15W

III. Software

1. Arduino IDE

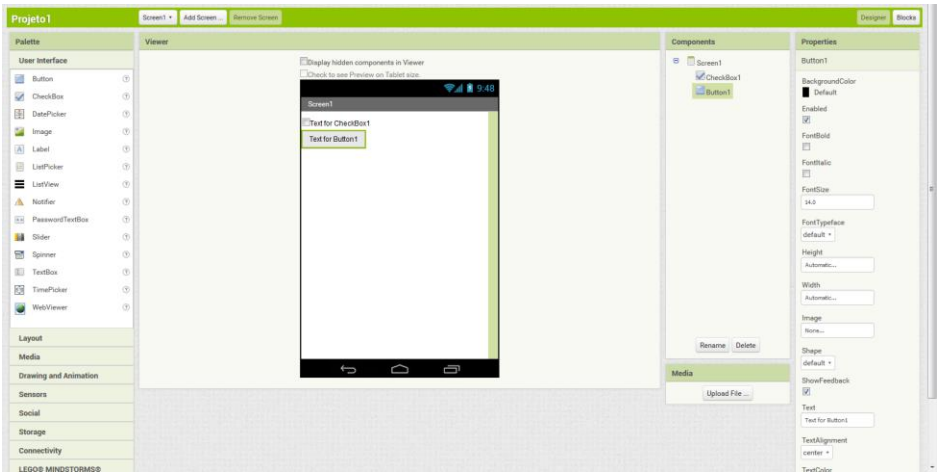
- The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.



Overview of Arduino IDE

2. MIT App Inventor

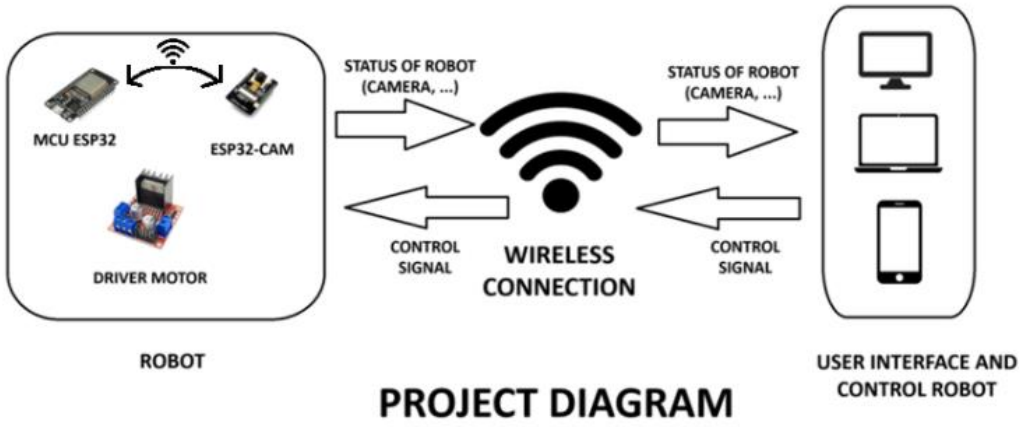
- MIT App Inventor (App Inventor or MIT AI2) is a high-level block-based visual programming language, originally built by Google and now maintained by the Massachusetts Institute of Technology. It allows newcomers to create computer applications for two operating systems: Android and iOS.



Overview of MIT App Inventor

IV. Implementation

1. Project Diagram:



- Users can use their smartphone, computer, … to control the ROBOT by CONTROL SIGNAL through WIRELESS CONNECTION.
- ESP 32 will receive the CONTROL SIGNAL which sent through WIRELESS CONNECTION and operate activities from CONTROL SIGNAL.
- ESP32 will act as a *server* and control robot, ESP32 CAM and USER DEVICES act as *clients* which connect to a server.
- The STATUS OF ROBOT (CAMERA, …) will be sent to USER DEVICE to help they can track the ROBOT.
- In this project, we will use HTTP Protocol with ESP32 Access Point Mode to make a WIRELESS CONNECTION between the ROBOT, CAMERA, and USER DEVICES. And the USER will use a DEVICE APP which we will create to control the ROBOT and receive video streamed from ESP32 CAM.

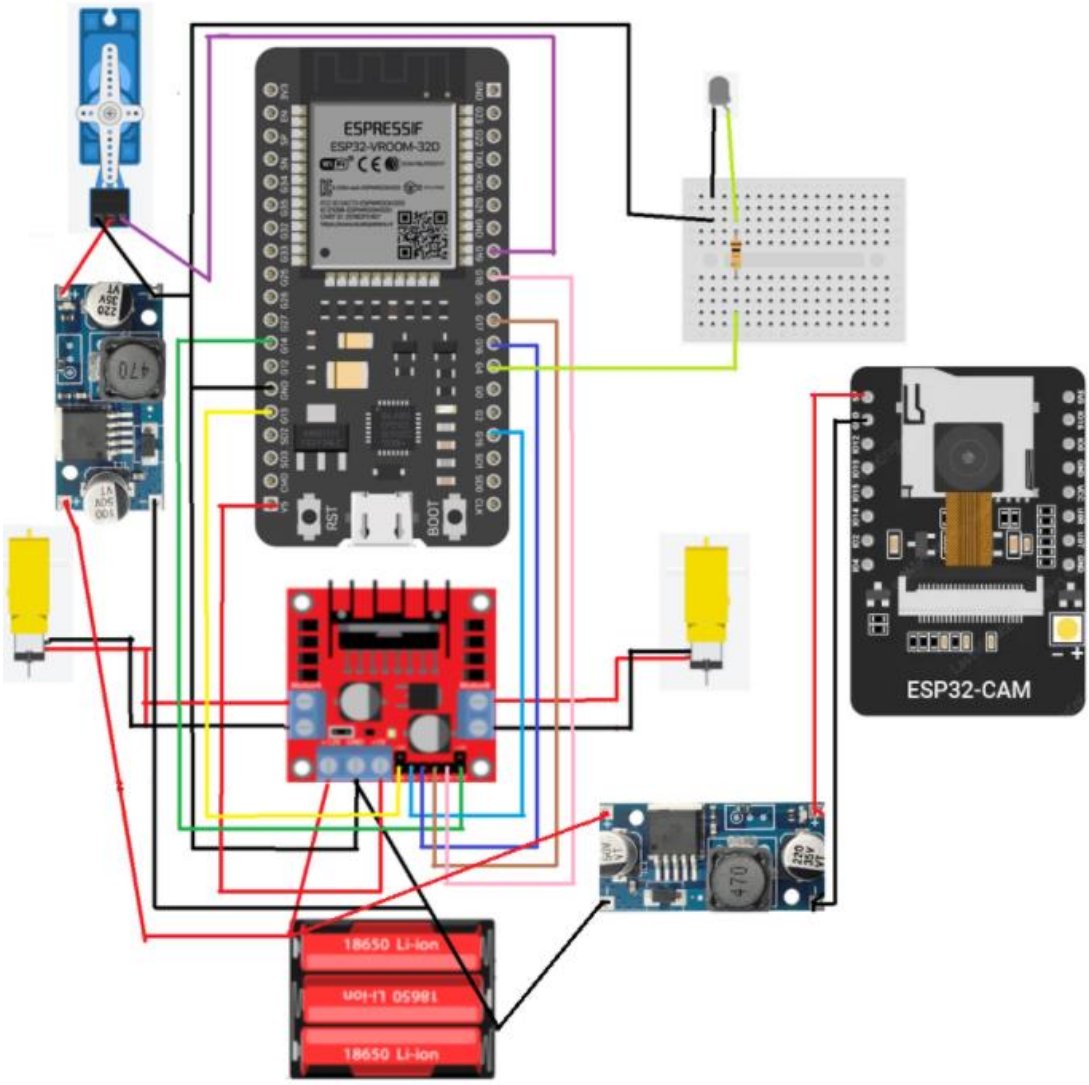
2. Robot Circuit:

2.1. Circuit

The table below shows connections between ESP32 GPIOs to other components of the ROBOT:

ENA (DC Motor Driver pin)	ESP32 GPIO 13
ENB (DC Motor Driver pin)	ESP32 GPIO 14
IN1 (DC Motor Driver pin)	ESP32 GPIO 15
IN2 (DC Motor Driver pin)	ESP32 GPIO 16
IN3 (DC Motor Driver pin)	ESP32 GPIO 17

IN4 (DC Motor Driver pin)	ESP32 GPIO 18
LED pin	ESP32 GPIO 4
Servo Motor pin	ESP32 GPIO 19



Robot Circuit

- We will use 3 18650 Li-on batteries to power the DC Motor Driver Module L298N and use 5V Supply Power pin of DC Motor Driver Module L298N to power the ESP32.
- The robot will have 2 motors, each of motor will be placed at the left and the right side of the robot.
- The robot also has an LED which is placed in the front of the robot.
- We also use 2 LM2596 module convert 12V to 5V to supply Servo motor and ESP32 CAM.

2.2. Electric consumption:

- According to the diagram above, we use 3 lithium batteries with each lithium battery having a capacity of 3Ah and the generated voltage is 3.7V. So, we will have a supply voltage for the whole robot that will be 3.7V x 3 = 11.1V.

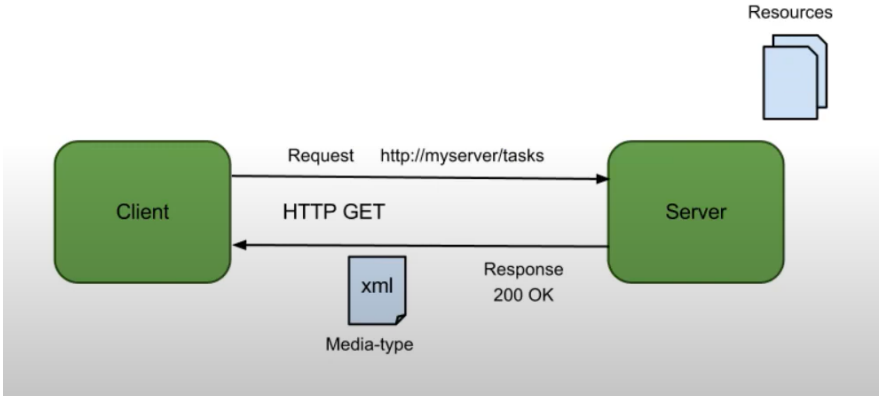
- The wattage of each LM2596 module is 15W, so the current consumption of the LM2596 will be about 15W / 11.1V = 1.35A. We use 2 LM2596 modules to power the ESP32 CAM and Servo motor, so the voltage consumption of the 2 circuits will be 1.35A x 2 = 2.7A.

- We use the DC Motor Driver Module L298N to power 2 motors and ESP32. The wattage of DC Motor Driver Module L298N is 25W, so the current consumption of the L298N will be about 25W/11.1V = 2.25A.

➔ Summarize, the current the robot needs to operate is about 2.25A + 2.7 A = 4.95 A (rounded up to 5V), and with 3 fully charged lithium batteries will have a capacity of about 9Ah. So, the maximum time the robot can be used before recharging the battery is 1.8 hours.

3. Software Implementation:

3.1. Network model.



Network connection between ESP32 as Server and Control Device, ESP32 CAM as Clients

3.2. Robot Software Implementation

As mentioned above, we will use Arduino IDE to program the ESP32 and ESP32 CAM.

3.2.1. ESP32 Programming Implementation

3.2.1.1. Define pins.

```
#define ena   13      // ENA ==> GPIO 13
#define enb   14      // ENB ==> GPIO 14
#define in1   15      // IN1 ==> GPIO 15
#define in2   16      // IN2 ==> GPIO 16
#define in3   17      // IN3 ==> GPIO 17
#define in4   18      // IN4 ==> GPIO 18
#define led 4         // LED ==> GPIO 4
#define servoPin 19 // Servo ==> GPIO 19
```

3.2.1.2. Initialization ESP32 Wi-Fi mode.

We use ESP32 Access Point mode with HTTP protocol to make a Wi-Fi connection between user devices, camera and the robot.

```
#include <WiFi.h>
#include <WebServer.h>
WiFiClient client;
WebServer server(80);

/* WIFI settings */
const char* ssid = "ESP WiFi";
const char* password = "12345678";
```

The ESP32 will have an access point whose name is “ESP WiFi” and password is “12345678”.

3.2.1.3. Set up the ESP32.

In this setup() function, we will set up all the components of the robot (Motor Driver, LED, Servo Motor, ESP32 Wi-Fi).

```
void setup()
{
  /* initialize motor control pins as output */
  pinMode(ena, OUTPUT);
  pinMode(enb, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(led, OUTPUT);
  MyServo.attach(servoPin);
  MyServo.write(90);
  Serial.begin(9600);
  //connectWiFi();
  WiFi.mode(WIFI_AP);           //Only Access point
  WiFi.softAP(ssid, password);  //Start HOTspot removing password will disable security
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
 // Starting WEB-server
  server.on ( "/", HTTP_handleRoot );
  server.onNotFound ( HTTP_handleRoot );
  server.begin();
}
```

3.2.1.4. Control robot's motors.

We use ESP32's GPIOs to control the Motor Driver Module. The control signal of Motor Driver Module is shown in the table below.

Input 1 (IN1 or IN3)	Input 2 (IN2 or IN4)	Spinning Direction
Low	Low	Motor OFF
High	Low	Forward
Low	High	Backward
High	High	Motor OFF

- Based on the signal table, we have built 5 main functions for the robot's motors including GO STRAIGHT, GO BACK, TURN LEFT, TURN RIGHT and STOP. ENA pin and ENB pin will be used to adjust the speed of the motors.

```
void STOP(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void BACKWARD(){
  analogWrite(ena, 100);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enb, 100);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
void TURN_LEFT(){
  analogWrite(ena, 0);
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  analogWrite(enb, 255);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
void TURN_RIGHT(){
  analogWrite(ena, 255);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enb, 0);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void FORWARD(){
  analogWrite(ena, 100);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enb, 100);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
```

3.2.1.5. Handle Root

- This function checks if the URL that the client requested from the Server has an argument "State" or not, and the Server will always send code 200 to the client.

```
void HTTP_handleRoot(void) {
  if( server.hasArg("State") ){
    Serial.println(server.arg("State"));
  }
  server.send ( 200, "text/html", "" );
  delay(1);
}
```

3.2.1.6. Control robot based on data.

- This function will operate the robot based on the data received from the application (client).

```
// Temp Variable
String  data =""; /* data received from application */
int ledState = 0;

void Control(String data){
  //Control LED
  if (data == "onled"){
     Serial.println("on");
     ledState = 1;
  }
  else if (data == "offled"){
    Serial.println("off");
    ledState = 0;
  }
  if (ledState == 1) digitalWrite(led, HIGH);
  else if (ledState == 0) digitalWrite(led, LOW);
  //Control Servo
  if (data == "0"){
      Serial.println("S0");
      MyServo.write(0);
  }
  else if (data == "1"){
    Serial.println("S1");
    MyServo.write(90);
  }
  else if (data == "2"){
    Serial.println("S2");
    MyServo.write(180);
  }
 // Control Motors
 if (data == "forward"){
   Serial.print("F");
   Serial.println(ledState);
   FORWARD();
 }
 else if (data == "backward"){
   Serial.print("B");
   Serial.println(ledState);
   BACKWARD();
 }
```

```
else if (data == "left"){
  Serial.print("L");
  Serial.println(ledState);
  TURN_LEFT();
}
else if (data == "right"){
  Serial.print("R");
  Serial.println(ledState);
  TURN_RIGHT();
}
else if(data == "stop"){
  Serial.print("S");
  Serial.println(ledState);
  STOP();
};
```

3.2.1.7. Loop

- The ESP32 always handles clients. And data variable will take the value of argument “State” which server received from client. And the robot will operate continuously according to the data variable.

```
void loop()
{
    server.handleClient();
    data = server.arg("State");
    Control(data);
}
```

3.2.2. ESP32 CAM software implementation.

To program the ESP32 CAM, we will use 3 libraries which have been built in the Arduino IDE to control the camera: *app_http.cpp*, *camera_index.h* and *camera_pins.h*

3.2.2.1. Include libraries and define pins.

```
#include "esp_camera.h"
#include <WebServer.h>
#include <WiFi.h>
WiFiServer server(80);

#define CAMERA_MODEL_AI_THINKER // Has PSRAM
#include "camera_pins.h"
```

3.2.2.2. Set up Wi-Fi to connect to server with a static IP address.

```
const char* ssid = "ESP WiFi";
const char* password = "12345678";
IPAddress local_IP(192, 168, 4, 6);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);
```

3.2.2.3. Setup ESP32 CAM

We will setup the ESP32 CAM by initializing the pins and initializing the camera and connect to the Wi-Fi which broadcasted from the server (ESP32).

- Config pins:

```
void setup() {
  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_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG; // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;
```

```
  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                     for larger pre-allocated frame buffer.
  if(config.pixel_format == PIXFORMAT_JPEG){
    if(psramFound()){
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif
```

- Init the camera:

```
  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
  }
#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif
#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif
```

- Connect to the Wi-Fi:

```
  WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS);
  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}
```

3.2.2.4. Loop function

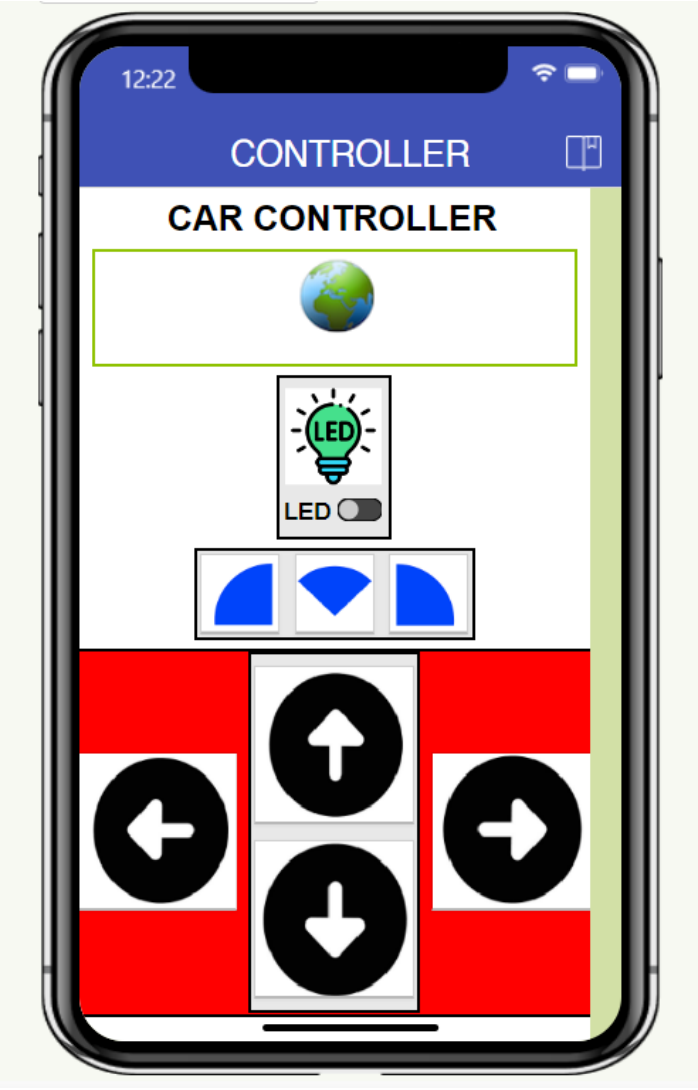
In the loop function, we do nothing.

```
void loop() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```

3.3. Robot Control App Implementation

3.3.1. Design the application's interface.

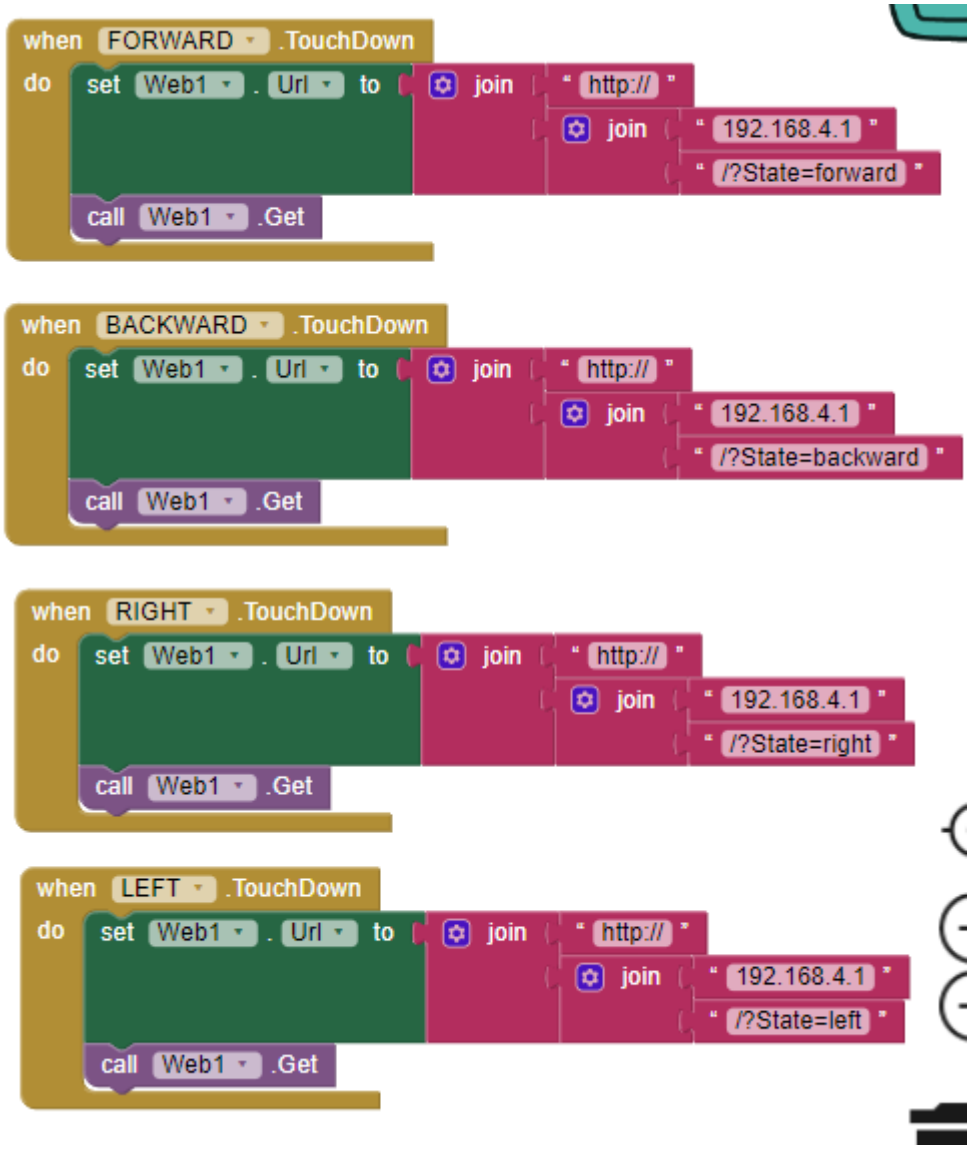
- We use a WebViewer object to access the video stream path of camera (http://192.168.4.6:81/stream) to view video streamed from the camera, 4 buttons to control the motors of the robot, 3 buttons to control the servo motor angle and use 1 switch to control the LED of the robot.

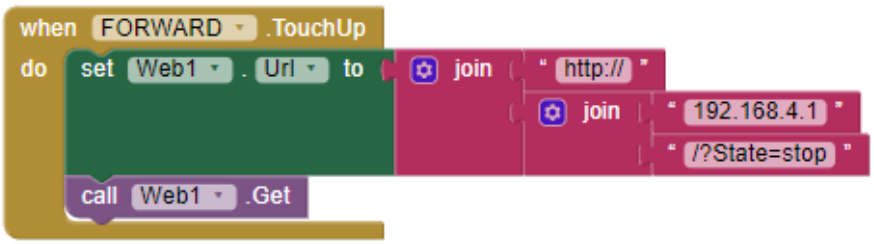


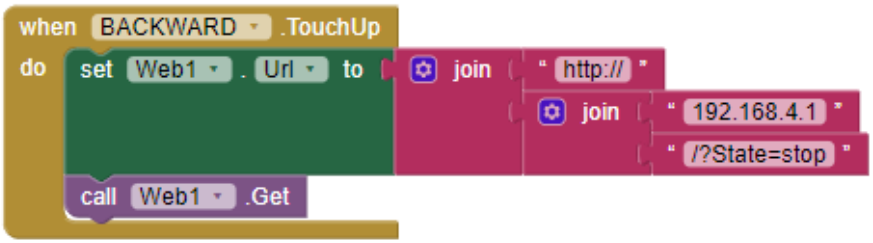
3.3.2. Design the backend of the application.

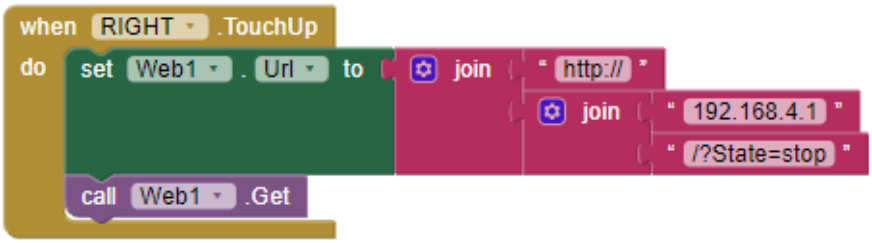
- For buttons, every time a button is pressed (Touch Down or Touch Up), the application will request objects in the Web Server using the GET method on the URL path on the Server (currently ESP32).

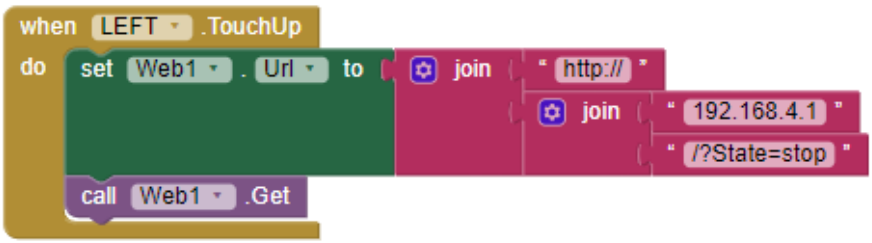
For example: When we press the "=>" button (Turn Right), the application will request the object using the GET method from the URL "192.168.4.1/?State=right", then when the Server (ESP32) receives the request from the application, it will control the robot to turn right

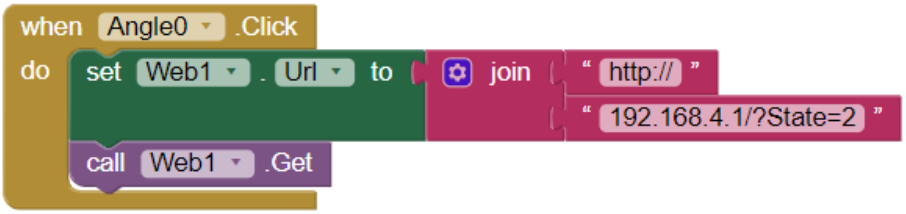


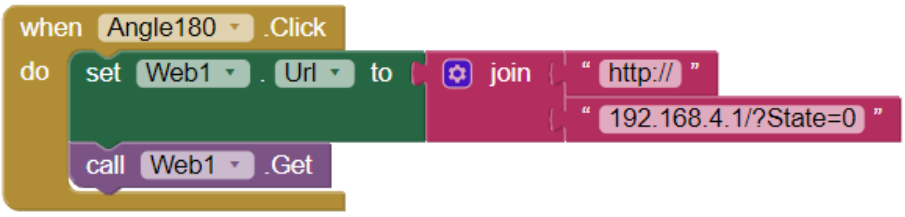

when FORWARD .TouchUp
do set Web1 . Url to join " http:// "
join " 192.168.4.1 "
" /?State=stop "
call Web1 .Get

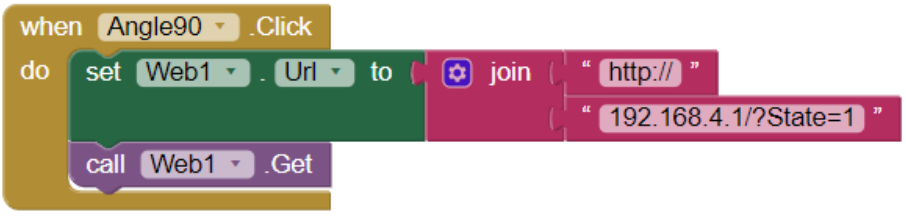

when BACKWARD .TouchUp
do set Web1 . Url to join " http:// "
join " 192.168.4.1 "
" /?State=stop "
call Web1 .Get


when RIGHT .TouchUp
do set Web1 . Url to join " http:// "
join " 192.168.4.1 "
" /?State=stop "
call Web1 .Get

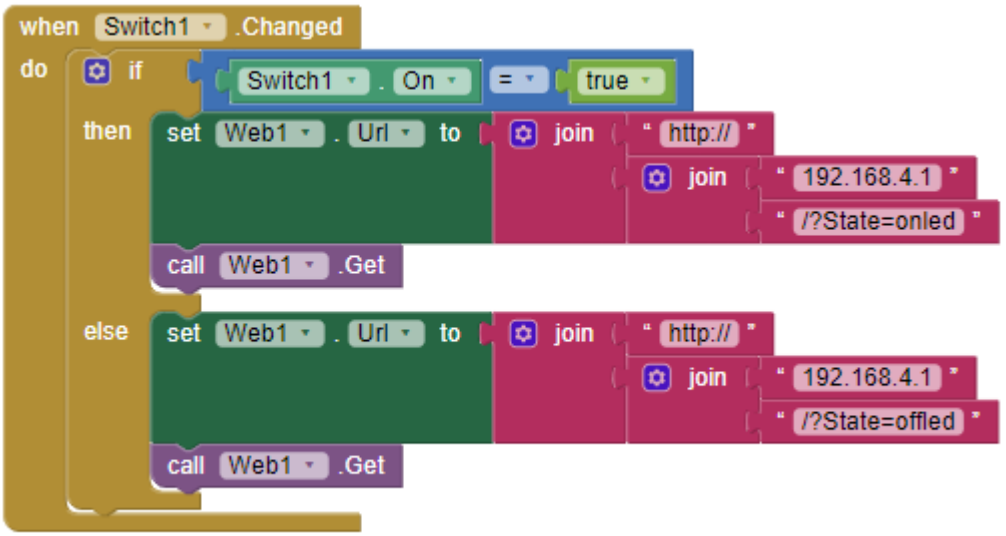

when LEFT .TouchUp
do set Web1 . Url to join " http:// "
join " 192.168.4.1 "
" /?State=stop "
call Web1 .Get


when Angle0 .Click
do set Web1 . Url to join " http:// "
" 192.168.4.1/?State=2 "
call Web1 .Get


when Angle180 .Click
do set Web1 . Url to join " http:// "
" 192.168.4.1/?State=0 "
call Web1 .Get

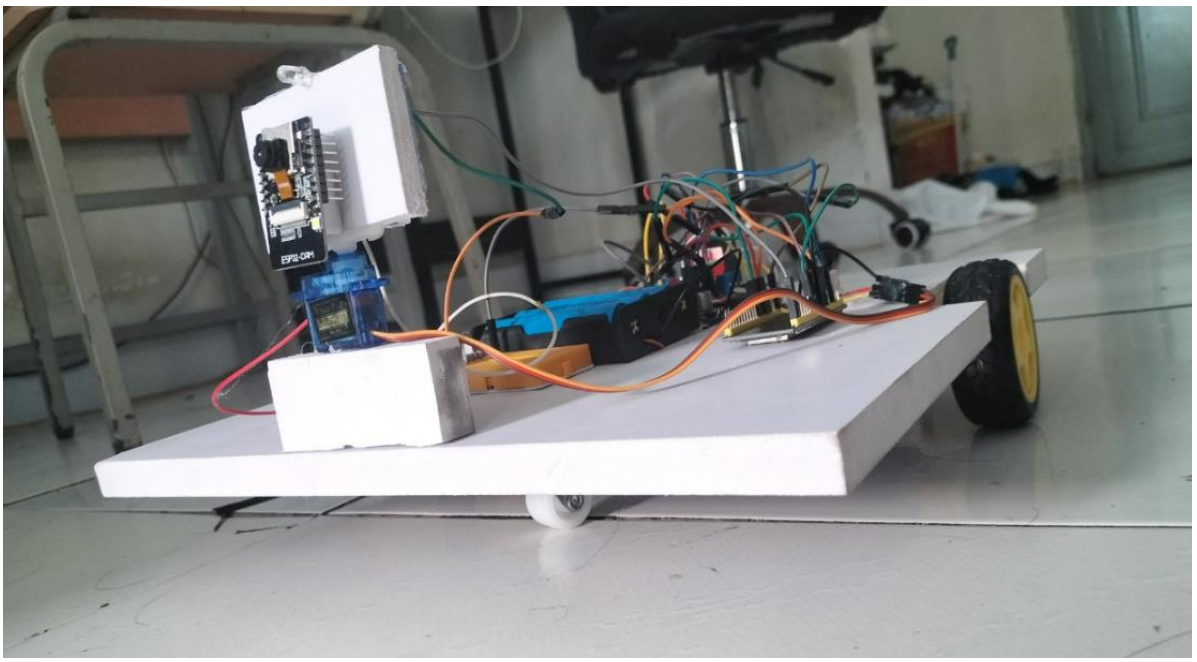

when Angle90 .Click
do set Web1 . Url to join " http:// "
" 192.168.4.1/?State=1 "
call Web1 .Get

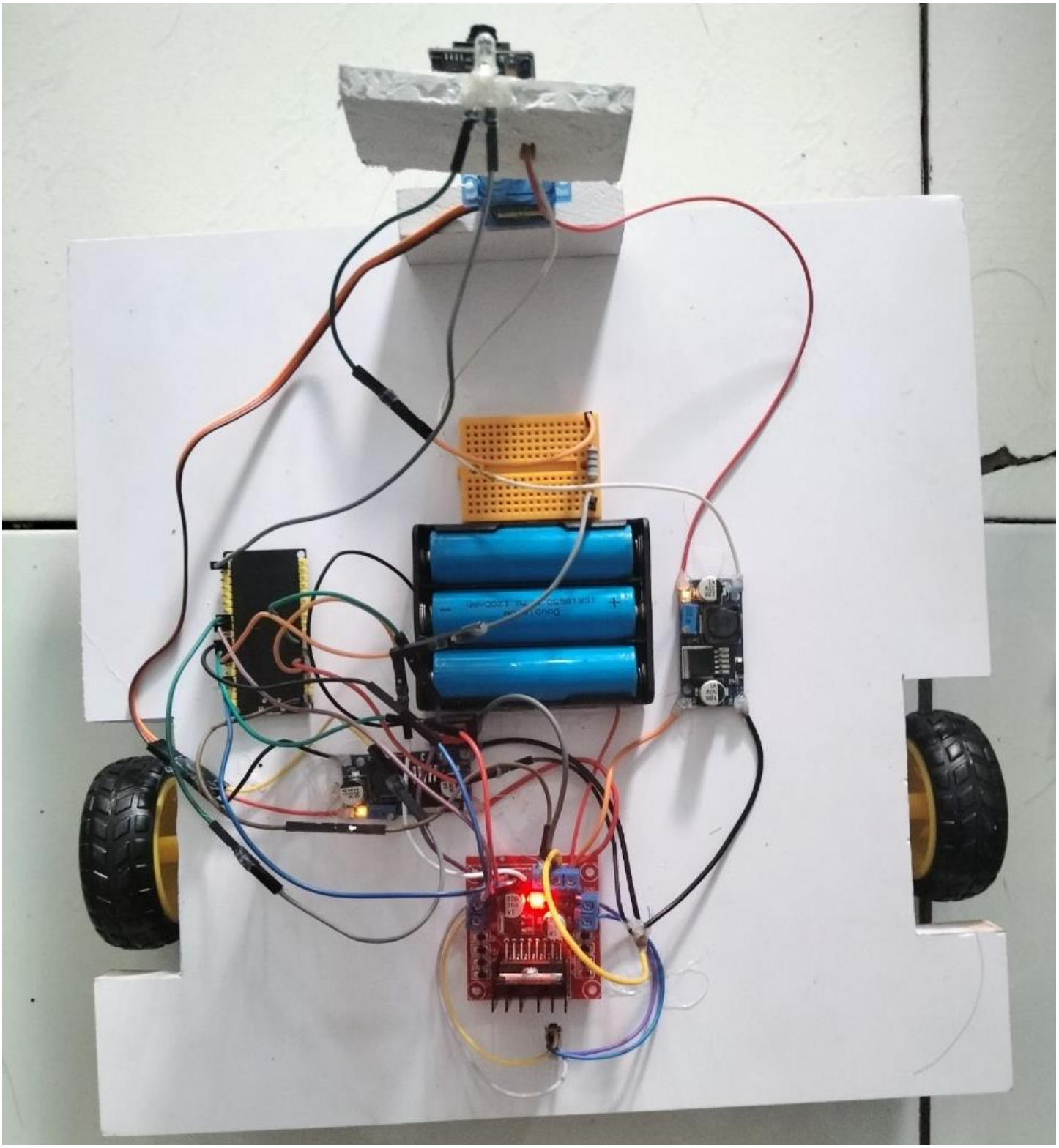
- Same for Switch, if switch is on, the LED of the robot will turn on and if switch is off, the LED of the robot will turn off.



4. Hardware Implementation

Based on the construction circuit diagram, we have built a realistic robot model.





V. Result

1. Achievement

- After assembling the robot and installing the control application on the phone, we connected to the ESP32 server, and we able to control the robot through the app.
- Here is a video below about our achievement we achieved with our project: *https://drive.google.com/file/d/1HVhV8xalqO2VKGNT3o4miHDFesW6CDvh/view?usp=drive_link*
- The robot's operating range is the ESP32's Wi-Fi operating distance (about a few dozen to a few hundred meters depending on the environment) from the robot to the control device.
- The camera is streamed in real time. Although the FPS is not high, and the image quality is not very good. However, the camera module we are using is a cheap module costing only 4$, so we cannot ask for higher requirements.

2. Difficulties

- Our limited knowledge of HTML/CSS/JavaScript, which is used to set up a personal website for us to control the robot. That's why we must use the MIT App Inventory application for ease to create a robot control application.
- In terms of security, this project is still not guaranteed well. Because if someone who is not the owner of the robot somehow gets the password of the robot's Wi-Fi network and has the robot control application. Then this person can take control of the robot owner and can control the robot.

3. Discussion

- We still need to improve a lot on our project. So below are some project goals that we hope to achieve in the future:
    - Using the MIT App Inventor application to control the robot can be replaced with a personal website that we program ourselves to control the robot.
    - Find a new way to increase the robot's security, perhaps allowing only 1 person to use the robot at a time.