GAZIANTEP UNIVERSITY

Bluetooth Controlled Farm Robot

EEE 499 GRADUATION PROJECT IN

ELECTRICAL & ELECTRONICS ENGINEERING

SUPERVISOR: ASSOC. PROF. DR. A. METE VURAL

BY

BURAK SEVİK

MEHMET ERCİN

MEHMET ZANA SÖNER

MÜCAHİT SAYİN

JANUARY 2022

ABSTRACT

Bluetooth Controlled Farm Robot

Graduation Project in Electrical and Electronics Engineering

Supervisor: ASSOC. PROF. DR. A. METE VURAL

January 2022, 36 pages

Modern agriculture has been changing the world for years as technology, engineering and agricultural biology develop exponentially.

People used oxen for plowing, for transport, and for powering machines that grind grain or supply irrigation. The lack of digitalization, engine power and automation in agriculture caused excessive use of manpower, decreased productivity, and time.

In this modern world we live in, we see that these primitive solutions are no longer valid. But modernized agriculture doesn't solve all the problems. In a world where resources are draining too fast, optimizing the modern agriculture is vital.

In this project, the main idea is to make a Bluetooth controlled farm robot that will make field work easier for farmers by its useful features. The robot can move forward, backward, left and right sides. It has a sensor to measure temperature and humidity of the environment. There is an IP camera on the front for remote observation of the plants and soil. In addition, the robot is powered by rechargeable batteries so it doesn't pollute the nature and doesn't contribute the climate change. With these advantages, farmers will be aware of the conditions of the fields and crops even if they are not physically in their fields.

Keywords: modern agriculture, automation in agriculture, robot, remote control

ÖZET

Bluetooth İle Kontrol Edilen Çiftlik Robotu

Mezuniyet Projesi, Elektrik ve Elektronik Mühendisliği Bölümü

Danışman: Doç. Dr. A. Mete VURAL

Ocak 2022, 36 sayfa

Modern tarım, teknoloji, mühendislik ve tarımsal biyolojinin katlanarak gelişmesiyle dünyayı değiştirmeye devam ediyor.

İnsanlar tarla sürmek, nakliye yapmak ve tahıl öğüten veya sulama sağlayan makinelere güç sağlamak için öküzleri kullandılar. Tarımda motor gücü ve otomasyon eksikliği, aşırı insan gücü kullanımına, üretkenliğin azalmasına ve fazla zaman kullanımına neden olmuştur.

Yaşadığımız bu modern dünyada, bu ilkel çözümlerin artık geçerli olmadığını görüyoruz. Ancak modernize edilmiş tarım tüm sorunları çözmüyor. Kaynakların çok hızlı tükendiği bir dünyada, modern tanımı optimize etmek hayati önem arz ediyor.

Bu projedeki ana fikir, teknolojik özellikler kullanarak çiftçilere yardım edecek bir Bluetooth ile kontrol edilen çiftlik robotu yapmaktır. Bu robot ileri, geri, sağa ve sola olmak üzere dört yöne hareket edebiliyor. Ön tarafında bulunan IP kamera ile tarladaki bitki ve toprağın uzaktan gözlemi yapılabilir. Ek olarak, robota güç vermek amacıyla yeniden şarj edilebilir piller kullanıldığı için çevreyi kirletmiyor ve iklim değişikliğine katkı sağlamıyor. Bu avantajlarla birlikte, çiftçiler tarlalarının ve ekinlerinin durumlarından orada bulunmasalar bile haberdar olacaklar.

Anahtar Kelimeler: modern tarım, tarımda otomasyon, robot, uzaktan kumanda

ACKNOWLEDGEMENTS

We would like to thank our supervisor, Assoc. Prof. Dr. A. Mete VURAL, who guided us during term.

We would like to acknowledge our family members for their financial and moral support to this project.

TABLE OF CONTENTS

LIST OF CHARTS

LIST OF FIGURES

1. INTRODUCTION

Modern agriculture is the combined product of digitalization and mechanization. Primitive tools for farming that needs manpower and animal power was replaced to more modern farm tools that are mainly powered by engines and hydraulics. For instance, scythe was used for mowing grass and harvesting crops for many years. We are now using harvesters and it’s way better than other hand tools. If these two products are compared in a job of harvesting a field of 4 hectares, 2 farmers will work with a scythe in this field and complete it in at least 3 days, while a combine harvester will complete it in about 45 minutes.

The competitive conditions that have developed as a result of the intense migration to the urban and the employment density in the service sector have pushed companies to digitalization studies, especially in the last 10 years. Since the agriculture sector is naturally a market of these digitalization studies, the use of digital and smart technologies in agriculture has started to increase gradually.

In this project, we are intended to make a robot car that reduces the time and effort spend on crops and fields for farmers. The main tasks of the robot are real time monitoring, and measuring temperature and humidity of the environment. It’s controlled by a mobile phone via Bluetooth. The robot has a micro controller (Arduino) for data processing, a servo motor to move left and right directions, a humidity and temperature sensor, a Bluetooth module for transmitting and receiving data, and an IP camera for real time monitoring.

The Bluetooth controlled farm robot has following features:

1) It can be controlled remotely with a mobile app via Bluetooth.
2) It can measure temperature and humidity.
3) Remote observation with IP camera.
4) Long-time use with rechargeable batteries.
5) In the mobile app, a triggering alarm value of the temperature or humidity can be set.
6) With the help of the virtual joystick in the mobile app, controlling the robot is smoother and easier.

The whole robot is powered by 3 series connected Li-Ion(maximum 12.6V) batteries, mainly for powering 2 DC motors at the rear and motor driver, and 9V alkaline battery. The 2 DC motors on the robot has the maximum torque of 800g/cm at 3V. These motors’ gears and outer shell is made of hard plastic so it helped with better weight distribution and lower cost. The servo for steering the robot car is connected to two axles, one for left wheel and the other for right wheel.

IP cameras are often used for house security but here in this project, it is used for observing field and crops. The live broadcasting process is very simple with this IP camera. It’s connected to a micro controller named ESP32 which has built-in features for video capturing such as; face recognition-detection, built-in camera flash, adjustable video resolution and image quality. This IP camera sending data via Wi-Fi so pairing the ESP32 micro controller with the mobile phone is important.

Since temperature and humidity are the two most important factors of the yield, it is vital to measure them with minimal error. The sensor that used in this project makes measurement per second and transmits the data to the Arduino. User can check the temperature and humidity values on the mobile app and, if desired, it is possible to set a trigger alarm for the desired humidity and temperature values in the app.

2. OPERATION OF THE SYSTEM

In order to make this farm robot, software and hardware integration is crucial. Choosing the right equipment for hardware that works correctly with the software we made is important.

In order to process the data sent from user using the mobile app in the most accurate way, the application and the Arduino code were constantly updated during the term.

2.1 Movement Control

The most important part of the robot making is the movement control. There are so many applications that have commonly 4DC motors, and four buttons for the direction and speed control in the mobile app. We thought having 4 buttons for direction control is not the best option. Instead of buttons and having 4DC motors, we designed and coded a virtual joystick for smoother and more accurate direction and speed control and used 2DC motor and a servo. User sends the axis data on the mobile app using the joystick via Bluetooth and Arduino process these data.

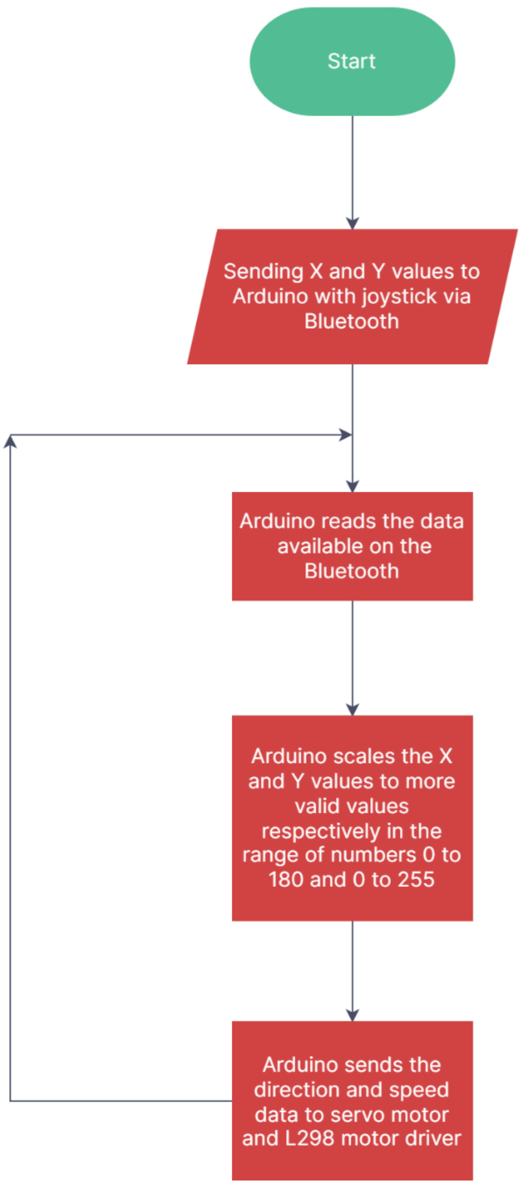

Start
Sending X and Y values to Arduino with joystick via Bluetooth
Arduino reads the data available on the Bluetooth
Arduino scales the X and Y values to more valid values respectively in the range of numbers 0 to 180 and 0 to 255
Arduino sends the direction and speed data to servo motor and L298 motor driver

Chart 1: Flowchart of motion control

2.2 Temperature and Humidity

Measuring the temperature and humidity is easily made by using a sensor named DHT11 that is capable of measuring both temperature and humidity at the same time at 1Hz frequency. It measures the data and send it to Arduino. Then, using a library called “DHT.h” and using Serial Communication these data is sent to mobile app via Bluetooth.

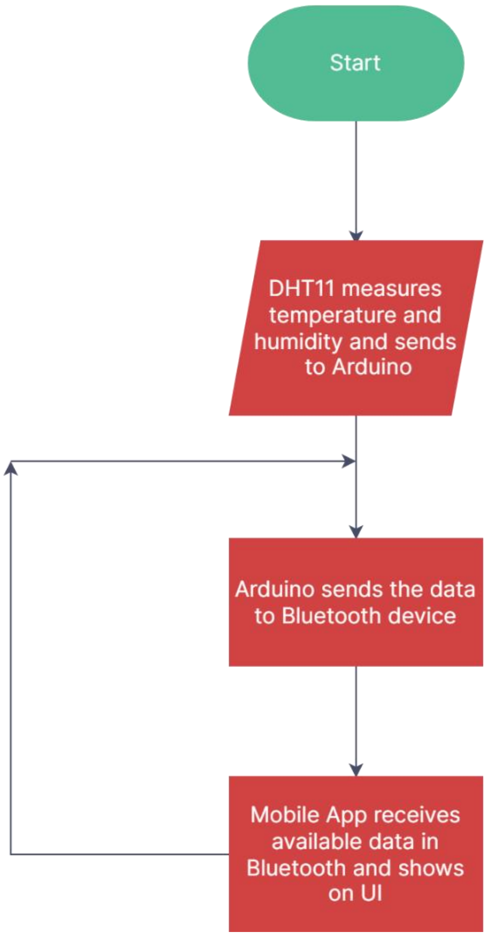

Start
DHT11 measures temperature and humidity and sends to Arduino
Arduino sends the data to Bluetooth device
Mobile App receives available data in Bluetooth and shows on UI

Chart 2: Flowchart of measuring temperature and humidity

2.3 Live Streaming

The solution for live streaming is to use an IP camera that uses Wi-Fi technology for sending and receiving data. IP camera and microcontroller used in this project is ESP-32 microcontroller. It is low cost, needs low power, has built-in antenna, flash lamp, and Wi-Fi technology.

Once it’s coded with Arduino, it gives an IP address to connect with any devices i.e. android, PC, and tablet. In the mobile app, there is a button. When pressed, an HTML page having this IP address is loaded on the screen.

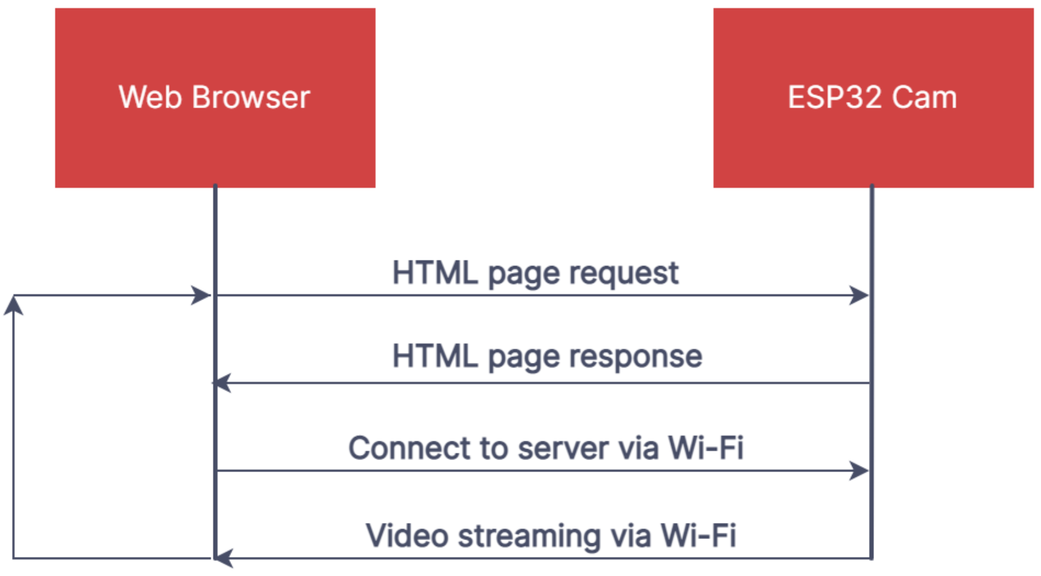


Chart 3: Flowchart of live broadcasting

3. HARDWARE

In this part, the hardware that is used in this project will be explained in details together.

3.1 ESP32 Cam

The ESP32-CAM is a very small ESP32-S chip with camera module. For connecting OV2640 camera and peripherals various GPIOs, as well as to store images captured with the camera or a file that can be used to store files to serve customers. There is also a micro SD card slot.

ESP32 can be used for mobile, wearable electronics and Internet of Things (IOT) applications.

Extremely low-power chip has many features such as fine-tuned clock switching, power modes, dynamic power scaling and live stream.

Though ESP-32’s Bluetooth and wifi communication facilities, it has the ability to wirelessly transfer the images taken from the camera. It has an antenna input that can increase the communication distance. It is a very suitable product for use in various remote image applications.

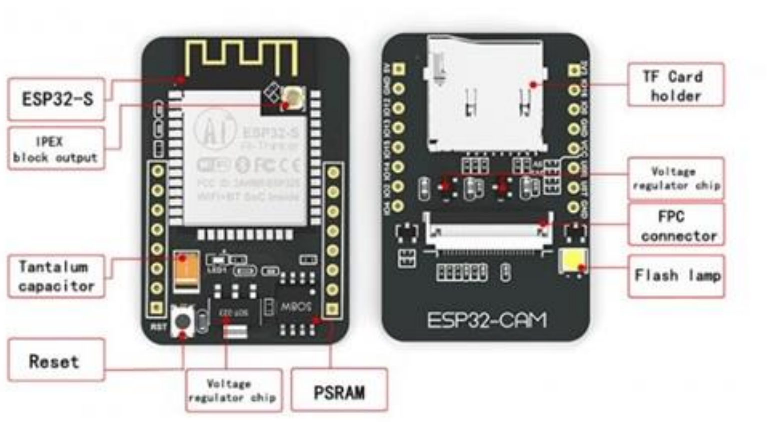


Figure 1: ESP32 pinouts

There are three GND pins and two pins for power: 3.3V or 5V. GPIO 1 and GPIO 3 are serial pins. To upload code to the module, we need these pins. In addition, GPIO 0 is also in flashing mode of ESP-32 plays an important role in determining. When GPIO 0 when connected to GND , ESP32 mode is in flashing. We disable ESP32's processor with GND. Because we load the code with Arduino.

3.2 DHT11

The DHT11 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data.

It measures the temperature between 0-50 °C with a +-2 °C accuracy and measures the humidity between 20-80% with a 5% accuracy.

- DHT11 has 8 bit microprocessor.
- 3.5v to 5v power supply
- 2.5mA current consumption
- digital output signal
- Temperature range from 0ºC to 50ºC
- Humidity can be measured between 20%RH and 80%RH

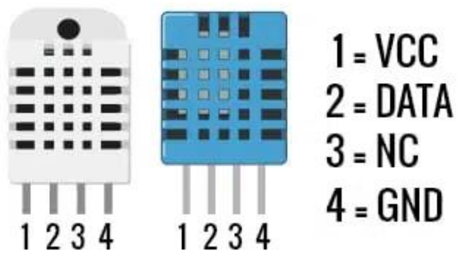


Figure 2: DHT11 sensor pinouts

1. VCC pin is connected to a pin having 5V on Arduino.
2. Data pin is connected to a digital pin on Arduino.
3. NC pin is not connected.
4. GND pin is connected to Arduino.

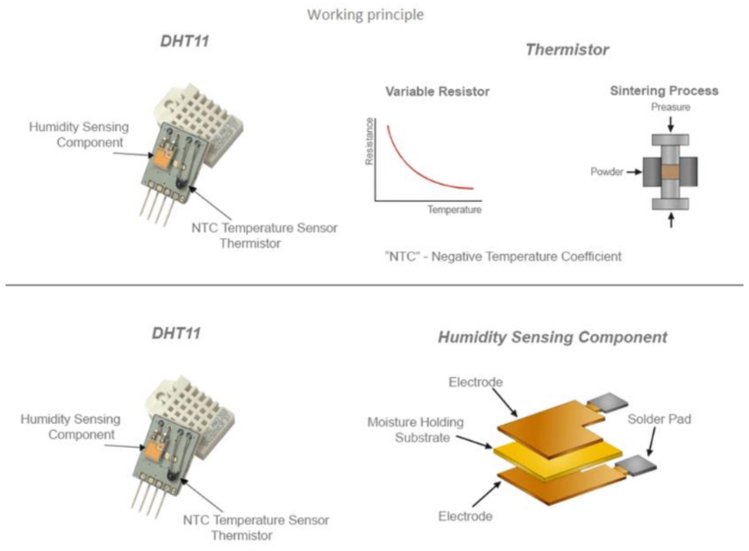

Working principle
DHT11
Humidity Sensing Component
NTC Temperature Sensor Thermistor
Thermistor
Variable Resistor
Resistance
Temperature
Sintering Process
Preasure
Powder
"NTC" - Negative Temperature Coefficient
DHT11
Humidity Sensing Component
NTC Temperature Sensor Thermistor
Humidity Sensing Component
Electrode
Moisture Holding Substrate
Solder Pad
Electrode

Figure 3: DHT11 interior parts

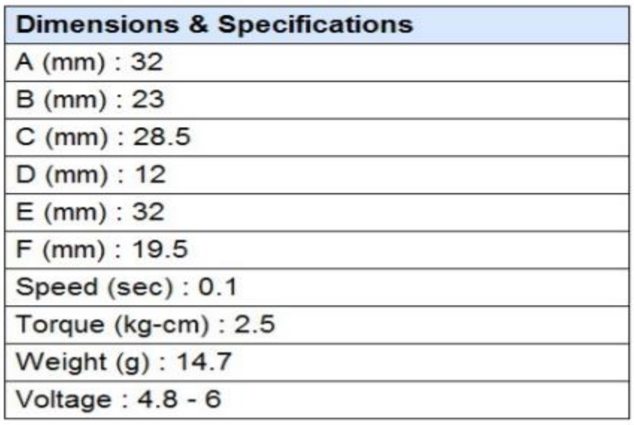
DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

3.3 Servo SG90

Servo is defined as a drive system that performs angular-linear position, speed and acceleration control in mechanisms without error. Servo (SG90) works with 4.8-6V voltage.

Figure 4: SG90 servo motor

Servo motors contain a DC motor that moves the motor. Apart from this motor, there is a gear mechanism, potentiometer and a motor driver circuit. The potentiometer measures the amount of rotation of the motor shaft. As the DC motor in the servo moves, the potentiometer rotates and the control circuit compares the position of the motor with the desired position and drives the motor. That is, servos work without the need for an external motor driver like other motors. The operating angles are limited to 180 degree.



Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

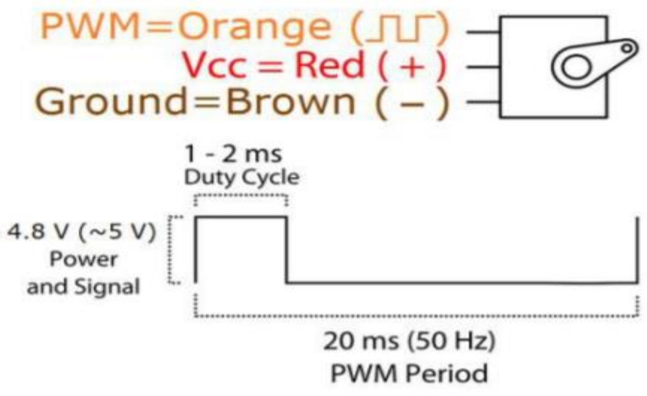


Figure 5: SG90 servo motor datasheet

Servo motors work with PWM (Signal Width Modulation) signals. These PWM signals can be provided from a microcontroller or remote control. The servo reads a pulse value every 20 ms. The pulse length determines the rotation of the motor. When the servos receive a command to move, they first move to the desired position and then stay in that position. Servos resist this power when an external force is applied to them while maintaining their position. They cannot hold their position indefinitely, and the pulse may need to be repeated to maintain their position. The pulse widths they need to move have minimums and maximums, and these values are variable.

3.4 DC Motor

A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

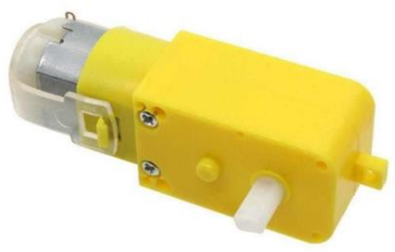
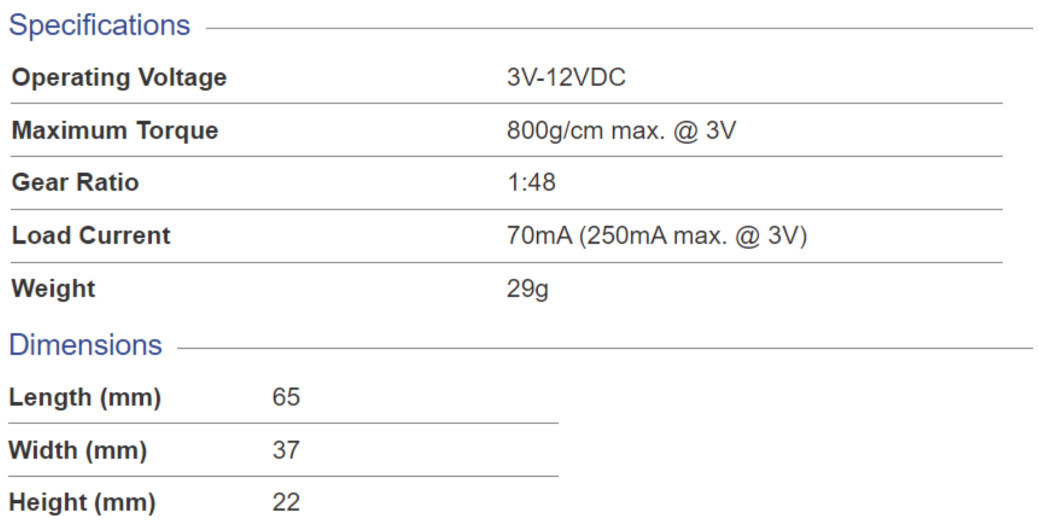


Figure 6: DC motor

DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight brushed motor used for portable power tools and appliances.

The DC motors used in this project is commonly used DC motor in robotics and hobby projects. It has a 1:48 gear ratio and maximum torque of 800g/cm at a minimum of 3V. Operating voltage is 3V-12VDC. Maximum load current of this motor is 250mA. These motors are powered by two series connected 3.8V Li-Ion batteries.



Specifications

Operating Voltage	3V-12VDC
Maximum Torque	800g/cm max. @ 3V
Gear Ratio	1:48
Load Current	70mA (250mA max. @ 3V)
Weight	29g

Dimensions

Length (mm)	65
Width (mm)	37
Height (mm)	22

Figure 7: Datasheet of the DC motor

3.5 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, 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 Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

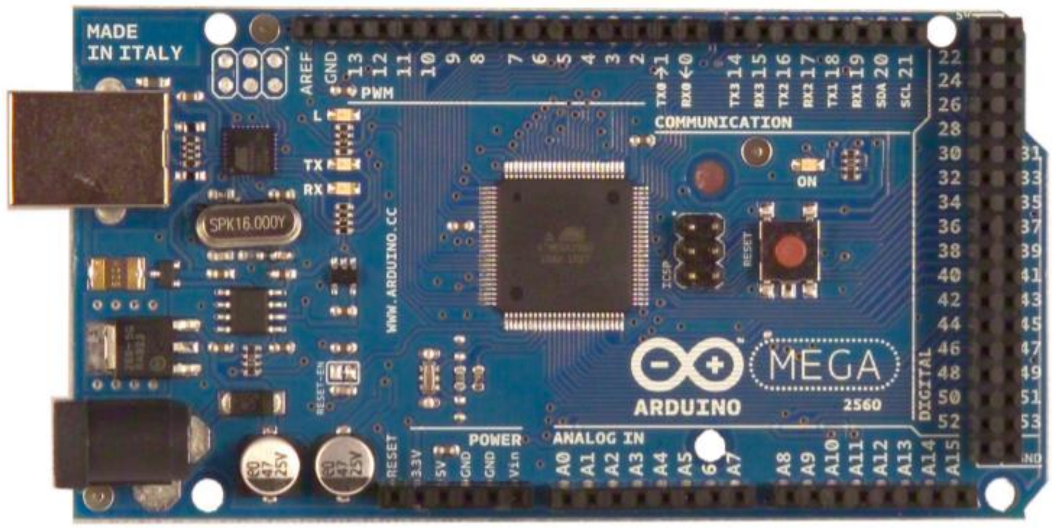
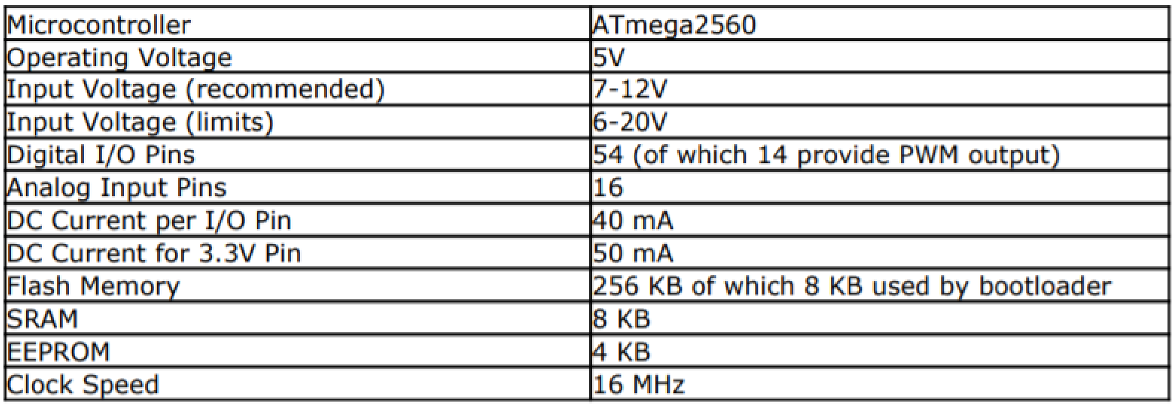


Figure 8: Arduino Mega 2560

Arduino Mega 2560 is selected in this project because

- It has extra 5V and GND pins compared to other Arduino boards e.g. Uno and Nano.
- It has sufficient digital PWM pins.
- Arduino IDE has a serial connection monitor that we used through entire project planning.
- It can easily communicate with a Bluetooth module via RX and TX pins.



Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Figure 9: Arduino Mega 2560 specs

3.6 Batteries

In this project, two series connected 18650 Li-Ion batteries and a 9V alcaline battery is used. Considering that the total power consumed by the robot is approximately 400-450 mAh, it is enough to use 1500mAh two series connected Li-Ion batteries for mainly powering the rear DC motors and motor driver board.

For powering the Arduino Mega 2560, a 9V alcaline battery is used.

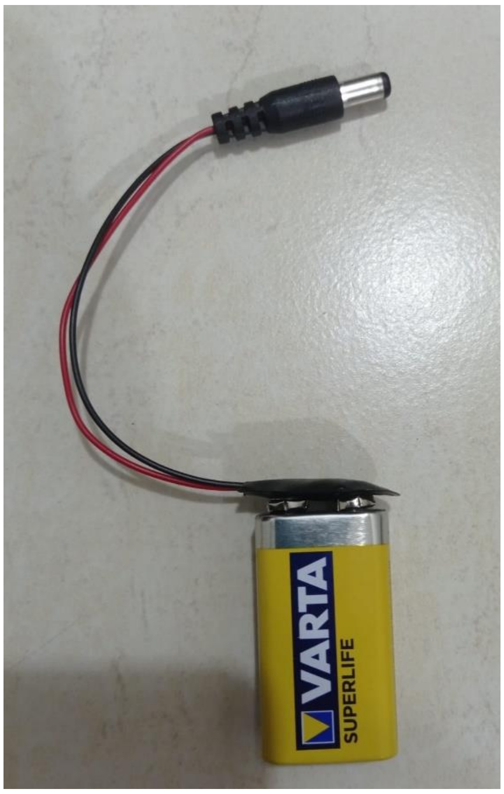


Figure 10: 9V alcaline battery

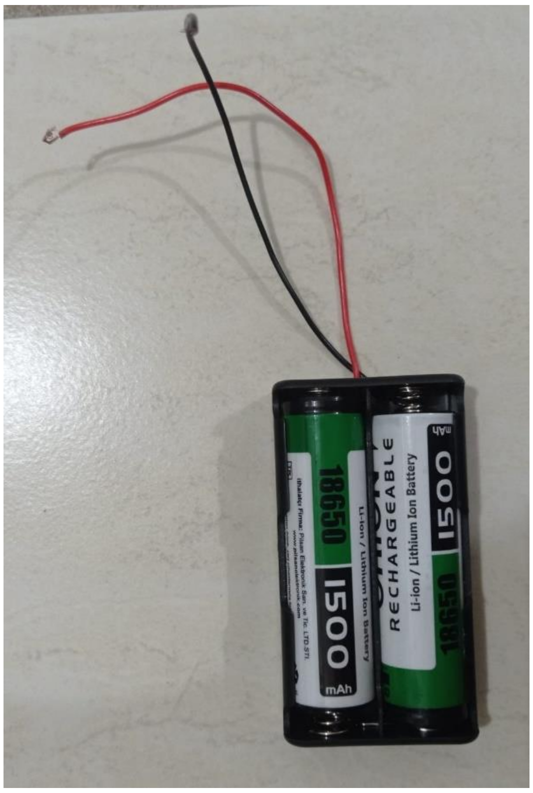


Figure 11: Li-Ion batteries

3.7 HC06

HC-06 is a Bluetooth module. In this project, we used it to receive data which is sent from mobile application by user and to send data from DHT to mobile application. Actually, HC-06 is like a data bridge. This module has 4 pins ,Vcc and GND pins used to supply the module. Rx pin is used to receive data. Tx pin used to transmit data. HC-06 module is used with Arduino together.

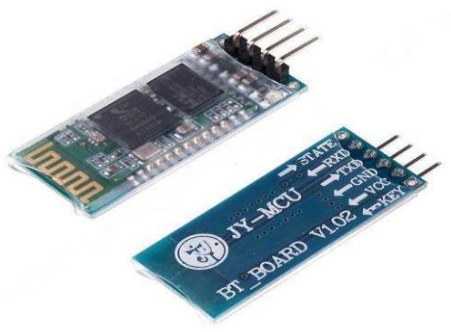


Figure 12: HC06 Bluetooth module

This module is very good for short distance communication between two microcontrollers or systems. It is very efficient for less than 100 meters working area. The device works at frequency range between 2.402 GHz and 2.280 GHz. It has operating voltage range from 3.3V to 6V and operating current of 40 mA. The module consumes very little power so it is very suitable for battery operated mobile systems. In addition it is very cheap and very easy to operate with all controllers and processors.

3.8 3D Printed Parts

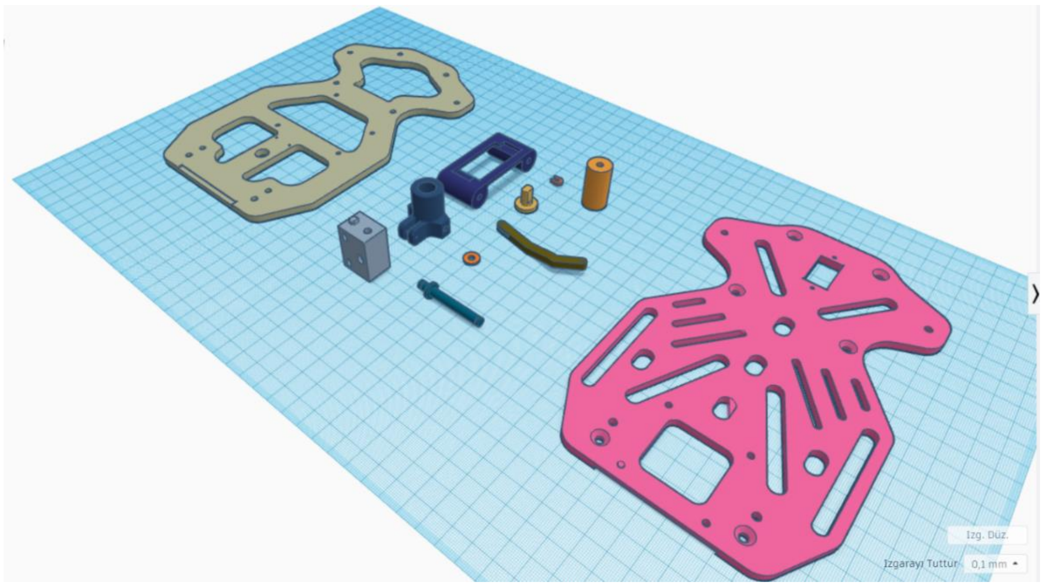


Figure 13: 3D parts used to build the robot

3.9 Final Product

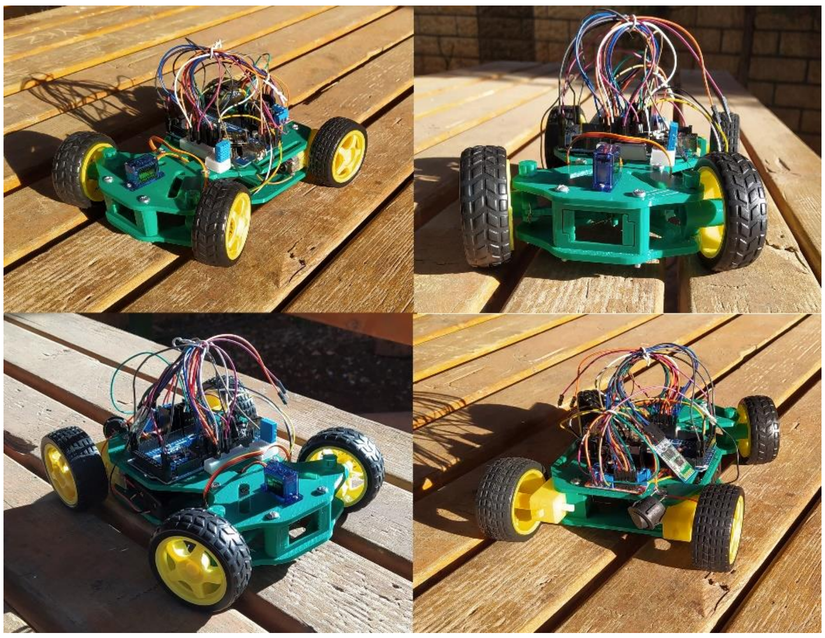


Figure 14: Final product

3.10 Circuit Schematic

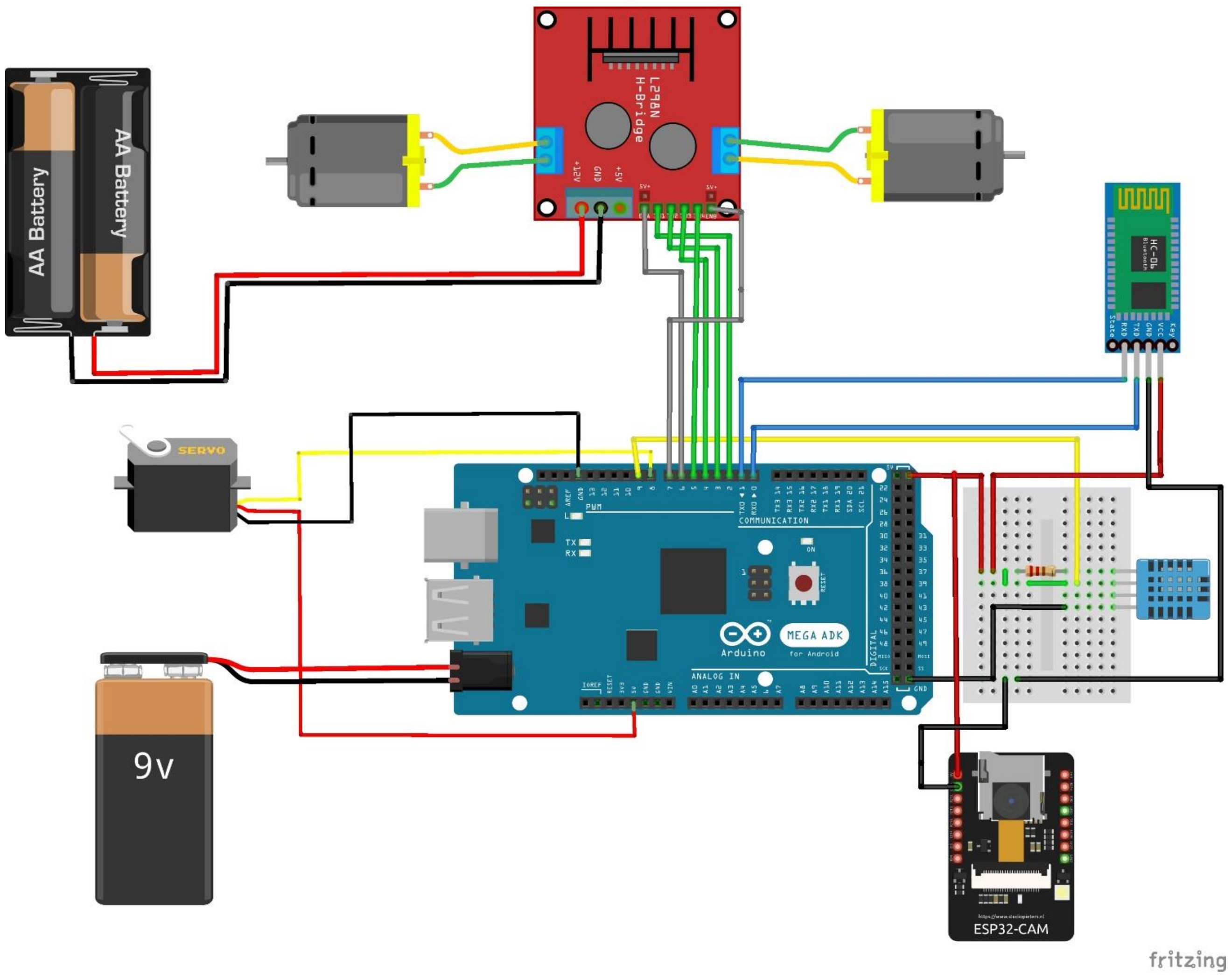


Figure 15: Circuit schematic of the project

4. SOFTWARE

There are three main codes in this project which are:

1. Main Arduino code for movement control of the robot and measuring temperature and humidity.
2. Mobile App code which is written in MIT App Inventor.
3. Arduino code to set ESP32-CAM for initial use.

4.1 Main Arduino Code

With the joystick in the application, we get X and Y coordinates data. We use the X data for direction control, and the Y data for motion control. The rotation capability of the servo that is used for directional control is 180 degrees. The limits of the X values that is sent from the joystick is values from 1 to 237. We scale these values from 0 to 180 in the Arduino code. However the values between 0 and 180 are not very accurate values because of our design. So we took the boundaries from 8 to 111. The Y coordinate values that is sent from joystick is between 1-237. In order to avoid confusion in the code, the data coming from the bottom of the y-axis of the joystick is in the range of 97-237. We change it in the range of 0 to 255. This part is our “backward” function. The values we get from the upper side of the y coordinate of the joystick is in the range of 1-97. We change it in the range of 255 to 0. This part becomes our “forward” function. And finally we get the (97,97) coordinates from the joystick for the “stop” function. We convert these values to the (0,0) coordinate in the code to send motor driver.

We read the data from the DHT11 sensor thanks to the library we included in the Ardunio code. this library allows us to use 2 special functions. These functions are dht.readHumidity() and dht.readTemperature().

```
#include "DHT.h"

#define DHTPIN 9

#define DHTTYPE DHT11

#include<Servo.h>

//SERVO SET

Servo myservo;

int servoPin=8 ;


//DHT SET

DHT dht(DHTPIN,DHTTYPE);

//millis()FOR DHT

unsigned long dht_previousMillis=0;
```

```
int interval_dht = 5000;

//JOYSTICK DATA
int yAxis;
int xAxis;
int mappedxAxis;
int mappedy1;
int mappedy2;

//millis() FOR JOYSTICK
unsigned long joy_previousMillis=0;
int interval_joy = 1000;

//L298 PINS
int dir11=2;
int dir12=3;
int dir21=4;
int dir22=5;
int speed1=6;
int speed2=7;

void setup() {
  Serial.begin(9600);
  myservo.attach(servoPin);
  dht.begin();
  pinMode(dir11,OUTPUT);
  pinMode(dir12,OUTPUT);
  pinMode(dir21,OUTPUT);
  pinMode(dir22,OUTPUT);
  pinMode(speed1,OUTPUT);
  pinMode(speed2,OUTPUT);
```

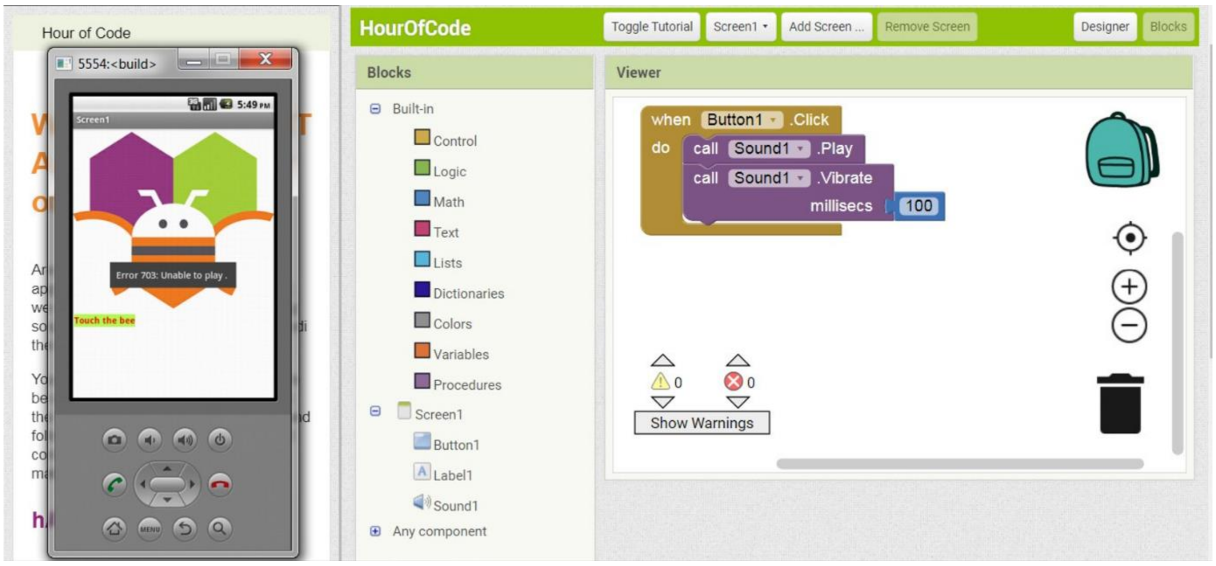
```
}


void loop() {
  //TEMP and HUM
  unsigned long currentMillis = millis();


   if((unsigned long)(currentMillis - dht_previousMillis)>=
interval_dht)
  {
    temp_and_hum();
    dht_previousMillis = millis();
  }
  char check;
  //JOYSTICK READ
  if(Serial.available()>=2)
{
  check = Serial.read();
  if(check=='x'){
     xAxis = Serial.read();
    }
  if(check=='y'){
     yAxis = Serial.read();
    }


}
 //JOYSTICK READINGS MAPPED
 mappedxAxis = map(xAxis,1,237,0,180);
 mappedy1 = map(yAxis,1,97,255,0);
 mappedy2 = map(yAxis,97,236,0,255);


 //JOYSTICK DELAY
```

```
  if((unsigned long)(currentMillis - joy_previousMillis)>=
interval_joy)
  {
    readxAxis();
    readyAxis();
    joy_previousMillis = millis();
    //CHECKING DATA IN SERIAL SCREEn
    Serial.print("CHAR: ");
    Serial.println(check);
    Serial.print("X: ");
    Serial.println(xAxis);
    Serial.print("Y: ");
    Serial.println(yAxis);

    }

  //SERVO WRITE
  if(mappedxAxis<=8)
  {
    mappedxAxis=8;
    myservo.write(8);
  }
  else if(mappedxAxis>8 && mappedxAxis<111)
  {

    myservo.write(mappedxAxis);
  }
  else if(mappedxAxis>=111)
  {
    mappedxAxis=111;
    myservo.write(111);
  }

  //DC MOTOR DIRECTION AND SPEED
  if(yAxis<97)
{
  ileri();
}
else if(yAxis>97){
  geri();
}
else if(yAxis=97){
  dur();
}
}

void temp_and_hum(){
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  Serial.print(t);
  Serial.print("C");
  Serial.print(";"); //Splitter
  Serial.print(h);
  Serial.println("%");
 }
void ileri(){
  digitalWrite(dir11,LOW);
  digitalWrite(dir12,HIGH);
  digitalWrite(dir21,LOW);
  digitalWrite(dir22,HIGH);
  analogWrite(speed1, mappedy1);
  analogWrite(speed2, mappedy1);

}

void geri(){

  digitalWrite(dir11,HIGH);

  digitalWrite(dir12,LOW);

  digitalWrite(dir21,HIGH);

  digitalWrite(dir22,LOW);

  analogWrite(speed1, mappedy2);

  analogWrite(speed2, mappedy2);

}

void dur(){

  digitalWrite(dir11,LOW);

  digitalWrite(dir12,LOW);

  digitalWrite(dir21,LOW);

  digitalWrite(dir22,LOW);

  analogWrite(speed1, 0);

  analogWrite(speed2, 0);

}
```

4.2 Mobile App

In this project, we used MIT App Inventor platform for coding and designing the mobile application. The reasons why we chose this platform to make the mobile app are:

- It is not hard to learn and doesn't require too much expertise in programming languages.
- It has built-in Bluetooth server and client, user-friendly UI elements, database, cloud based project editing and saving, and free user made extensions for complex projects.
- It uses blocks-based programming language that is easier to use and learn than other programming languages.
- It has real-time emulator that helps to check if the codes or design suitable.

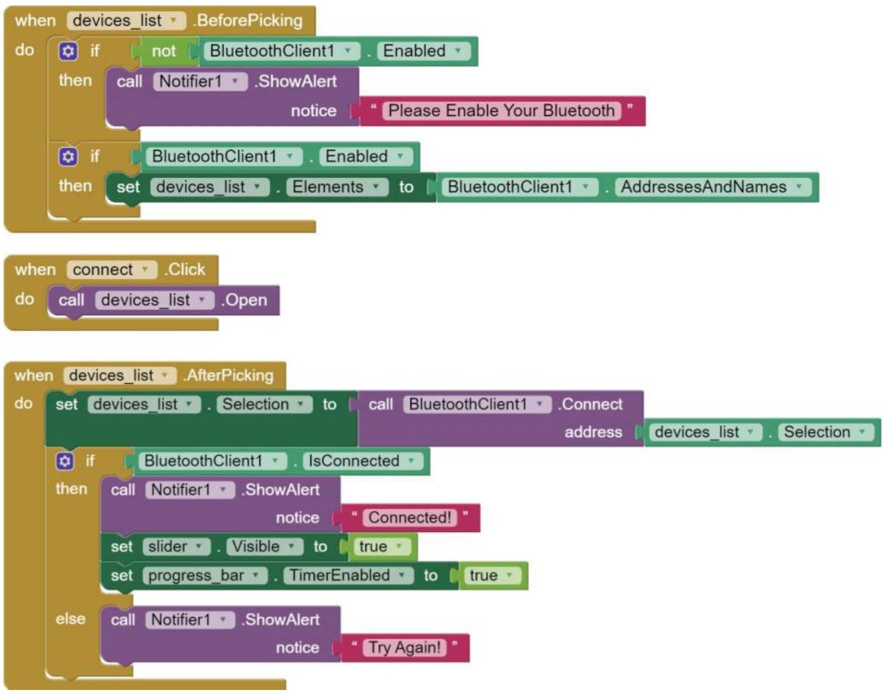


```
when Button1 .Click
do  call Sound1 .Play
    call Sound1 .Vibrate
                 millisecs 100
```

Figure 16: Example image of MIT app inventor workplace

The following images are the most important parts of the codes for mobile app.

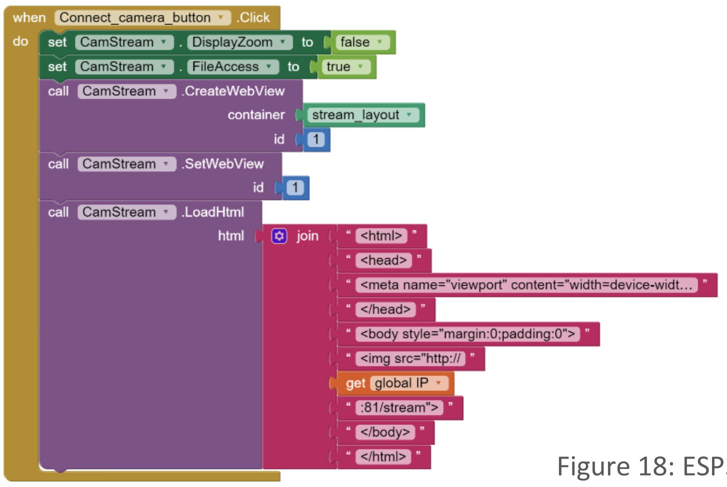
Bluetooth Connection



```
when devices_list .BeforePicking
do  if   not BluetoothClient1 . Enabled
    then call Notifier1 .ShowAlert
                        notice " Please Enable Your Bluetooth "
    if   BluetoothClient1 . Enabled
    then set devices_list . Elements to BluetoothClient1 . AddressesAndNames

when connect .Click
do  call devices_list .Open

when devices_list .AfterPicking
do  set devices_list . Selection to call BluetoothClient1 .Connect
                                                     address devices_list . Selection
    if   BluetoothClient1 . IsConnected
    then call Notifier1 .ShowAlert
                        notice " Connected! "
         set slider . Visible to true
         set progress_bar . TimerEnabled to true
    else call Notifier1 .ShowAlert
                        notice " Try Again! "
```

Figure 17: Bluetooth connection block-code

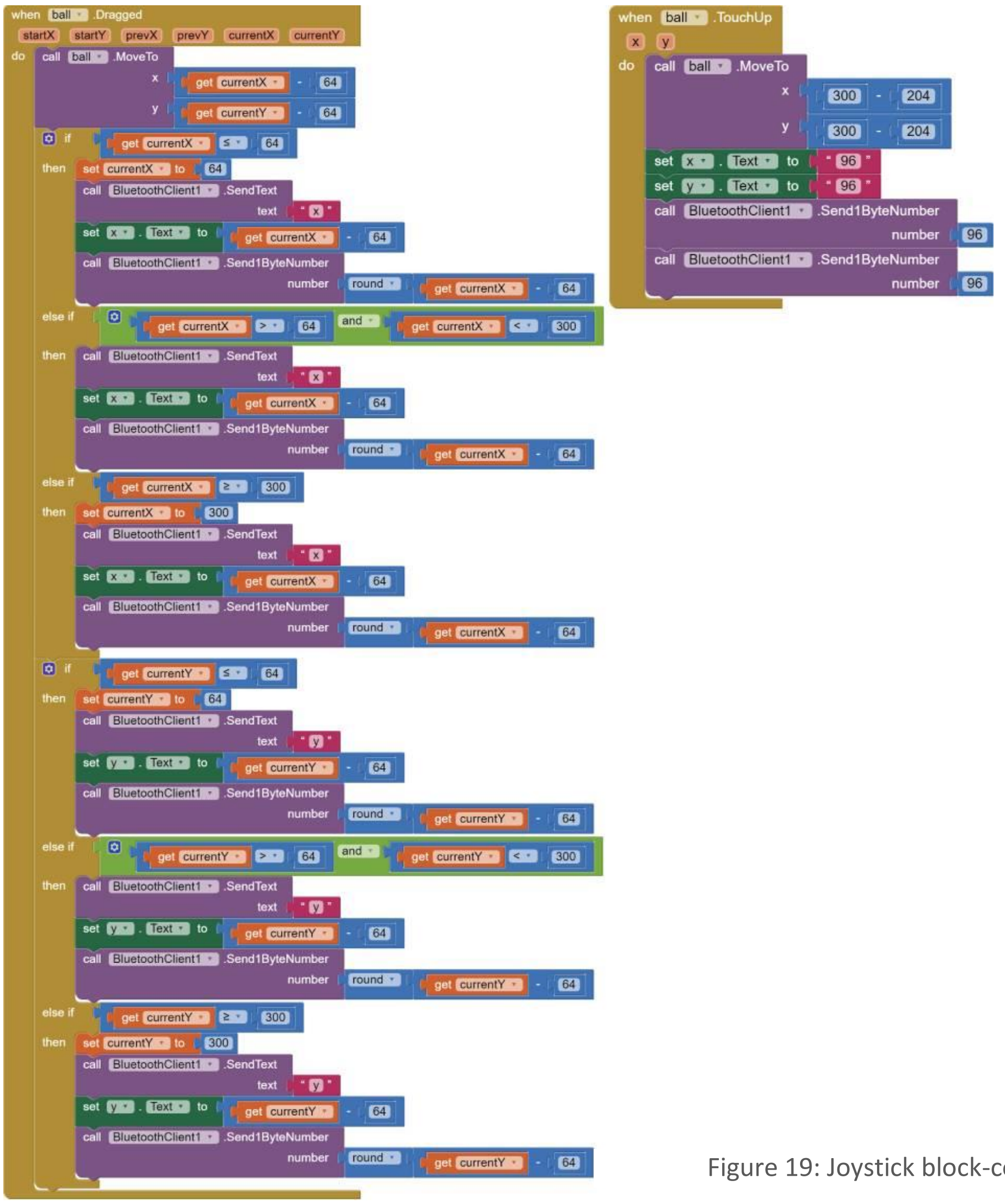
ESP32 IP Address Connection



```
when Connect_camera_button .Click
do  set CamStream . DisplayZoom to false
    set CamStream . FileAccess to true
    call CamStream .CreateWebView
                     container  stream_layout
                            id  1
    call CamStream .SetWebView
                            id  1
    call CamStream .LoadHtml
                          html  join  " <html> "
                                      " <head> "
                                      " <meta name="viewport" content="width=device-widt... "
                                      " </head> "
                                      " <body style="margin:0;padding:0"> "
                                      " <img src="http:// "
                                      get global IP
                                      " :81/stream"> "
                                      " </body> "
                                      " </html> "
```

Figure 18: ESP32 connection block-code

Joystick

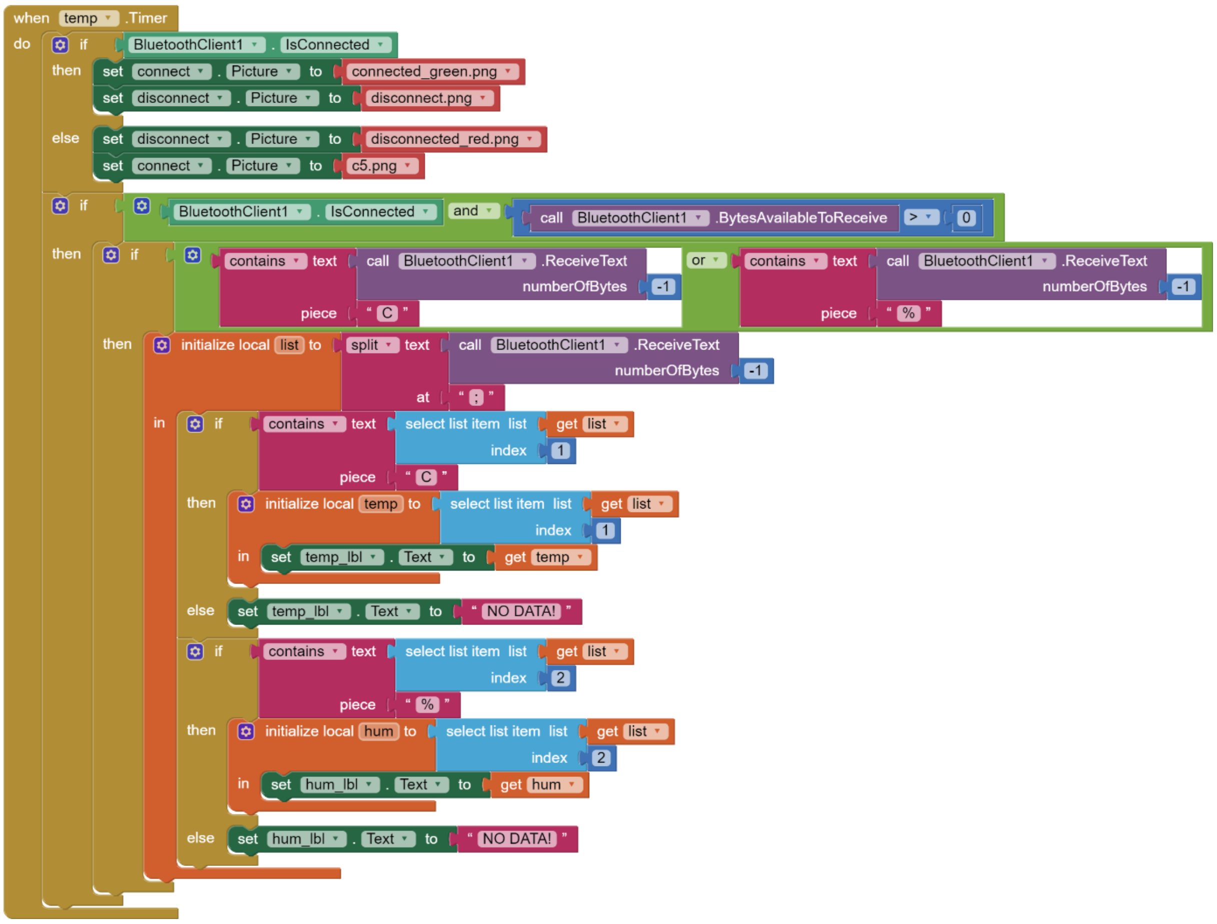


```
when ball .Dragged
  startX  startY  prevX  prevY  currentX  currentY
do  call ball .MoveTo
                x  get currentX - 64
                y  get currentY - 64
    if    get currentX ≤ 64
    then  set currentX to 64
          call BluetoothClient1 .SendText
                           text  " x "
          set x . Text to  get currentX - 64
          call BluetoothClient1 .Send1ByteNumber
                         number  round  get currentX - 64
    else if  get currentX > 64  and  get currentX < 300
    then  call BluetoothClient1 .SendText
                           text  " x "
          set x . Text to  get currentX - 64
          call BluetoothClient1 .Send1ByteNumber
                         number  round  get currentX - 64
    else if  get currentX ≥ 300
    then  set currentX to 300
          call BluetoothClient1 .SendText
                           text  " x "
          set x . Text to  get currentX - 64
          call BluetoothClient1 .Send1ByteNumber
                         number  round  get currentX - 64
    if    get currentY ≤ 64
    then  set currentY to 64
          call BluetoothClient1 .SendText
                           text  " y "
          set y . Text to  get currentY - 64
          call BluetoothClient1 .Send1ByteNumber
                         number  round  get currentY - 64
    else if  get currentY > 64  and  get currentY < 300
    then  call BluetoothClient1 .SendText
                           text  " y "
          set y . Text to  get currentY - 64
          call BluetoothClient1 .Send1ByteNumber
                         number  round  get currentY - 64
    else if  get currentY ≥ 300
    then  set currentY to 300
          call BluetoothClient1 .SendText
                           text  " y "
          set y . Text to  get currentY - 64
          call BluetoothClient1 .Send1ByteNumber
                         number  round  get currentY - 64
```

```
when ball .TouchUp
  x  y
do  call ball .MoveTo
                x  300 - 204
                y  300 - 204
    set x . Text to " 96 "
    set y . Text to " 96 "
    call BluetoothClient1 .Send1ByteNumber
                   number  96
    call BluetoothClient1 .Send1ByteNumber
                   number  96
```

Figure 19: Joystick block-code

Temperature and Humidity



```
when temp.Timer
do  if   BluetoothClient1.IsConnected
    then set connect.Picture to "connected_green.png"
         set disconnect.Picture to "disconnect.png"
    else set disconnect.Picture to "disconnected_red.png"
         set connect.Picture to "c5.png"
    if   BluetoothClient1.IsConnected and call BluetoothClient1.BytesAvailableToReceive > 0
    then if   contains text call BluetoothClient1.ReceiveText numberOfBytes -1
                       piece "C"
              or contains text call BluetoothClient1.ReceiveText numberOfBytes -1
                       piece "%"
         then initialize local list to split text call BluetoothClient1.ReceiveText numberOfBytes -1
                                               at ";"
              in if   contains text select list item list get list
                                                     index 1
                               piece "C"
                 then initialize local temp to select list item list get list
                                                                 index 1
                      in set temp_lbl.Text to get temp
                 else set temp_lbl.Text to "NO DATA!"
                 if   contains text select list item list get list
                                                     index 2
                               piece "%"
                 then initialize local hum to select list item list get list
                                                                index 2
                      in set hum_lbl.Text to get hum
                 else set hum_lbl.Text to "NO DATA!"
```

Figure 20: DHT11 block-code

The following images are the design part of the mobile app.

<u>Opening Screen</u>

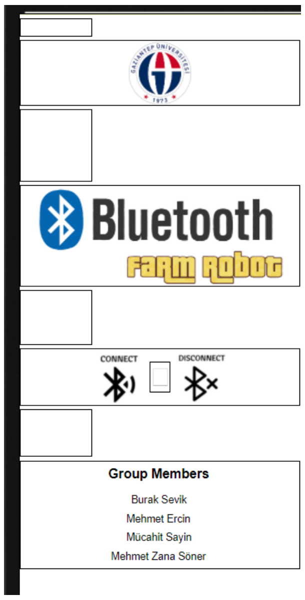


Figure 21: Opening screen

<u>Main Screen</u>

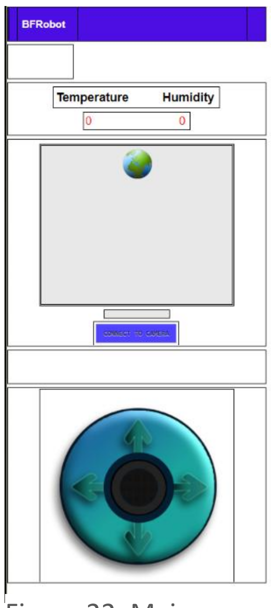


Figure 22: Main screen

<u>Menu Screen</u>

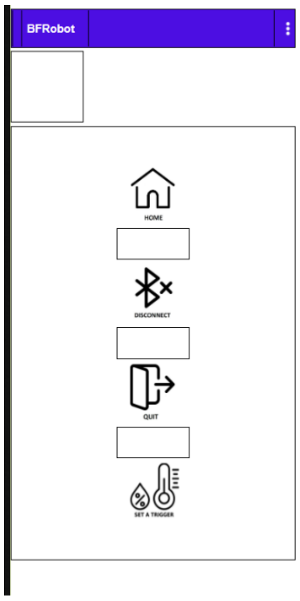


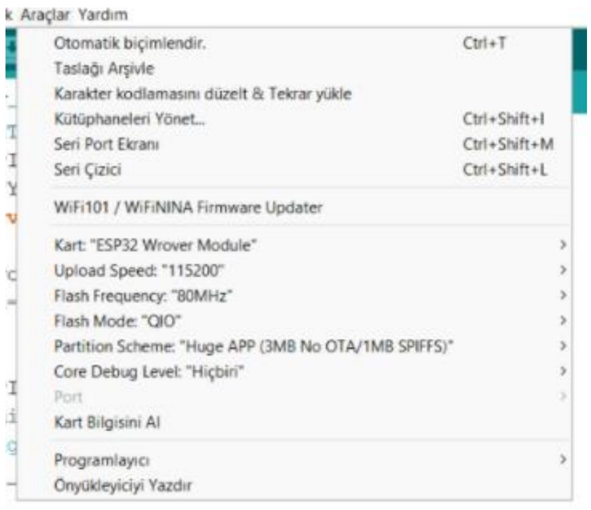
Figure 23: Menu screen

4.3 ESP32 Cam Arduino Code

ESP-32 is a Wi-Fi and Bluetooth development board. It has its own processor on it. Even without using Arduino, good projects can be made on ESP-32, but we will throw the code with Arduino in our Farm Robot project. We will short-circuit the processor of ESP-32 so that ESP-32 and Arduino do not get damaged while throwing the code. After uploading the code, it will be enough to get the IP address from the serial port screen for live broadcast and then just feed the ESP-32 module with power

We will be able to control our car with a real-time camera after we paste the IP address that we will stream into the browser section at the top of the vehicle control screen of our application. It took us quite a while to load the ESP-32 code and draw the circuit.

It gave a lot of errors, uploaded in a very long time, we reached the solutions of the errors a little late due to limited resources but eventually we solved this problem, and we succeeded live broadcast with ESP-32. In order to encode ESP-32, the settings in the tools section of Arduino IDE should be as follows.


k Araçlar Yardım

Otomatik biçimlendir. Ctrl+T
Taslağı Arşivle
Karakter kodlamasını düzelt & Tekrar yükle
Kütüphaneleri Yönet... Ctrl+Shift+I
Seri Port Ekranı Ctrl+Shift+M
Seri Çizici Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Kart: "ESP32 Wrover Module"
Upload Speed: "115200"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Partition Scheme: "Huge APP (3MB No OTA/1MB SPIFFS)"
Core Debug Level: "Hiçbiri"
Port
Kart Bilgisini Al
Programlayıcı
Önyükleyiciyi Yazdır

Figure 24: Options menu in Arduino IDE

We got this code from the sample codes for ESP-32 of Arduino IDE. After sending this code to Arduino once, we just power the ESP-32. The code of ESP-32 has to be different from the code of the vehicle because ESP-32 has its own processor and a card like Arduino.

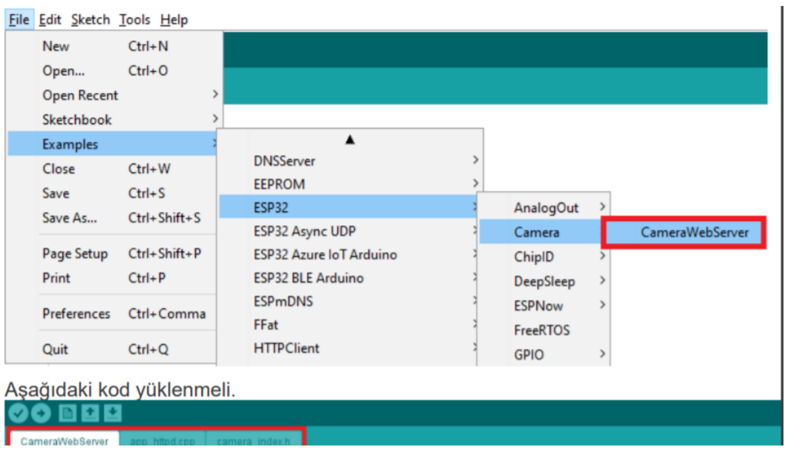


Figure 25: Example window of CameraWebServer

```
#include "esp_camera.h"
#include <WiFi.h>
//select camera type
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"
// enter Wi-Fi name and password
const char* ssid = "wifi";
const char* password = "password";

 void startCameraServer();
 void setup() {

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();
  pinMode(8,OUTPUT);

  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;

  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    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);
    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 blightness just a bit
    s->set_saturation(s, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);
  WiFi.begin(ssid, password);

  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");
}
void loop() {
  digitalWrite(8,HIGH);
  delay(500);
  digitalWrite(8,LOW);
  delay(500);
  }
```

After coding the ESP-32 with Arduino, it is enough to remove the GPIO 0 pin from the GND and press the reset button on the ESP-32 to get the ip address from the serial port screen.

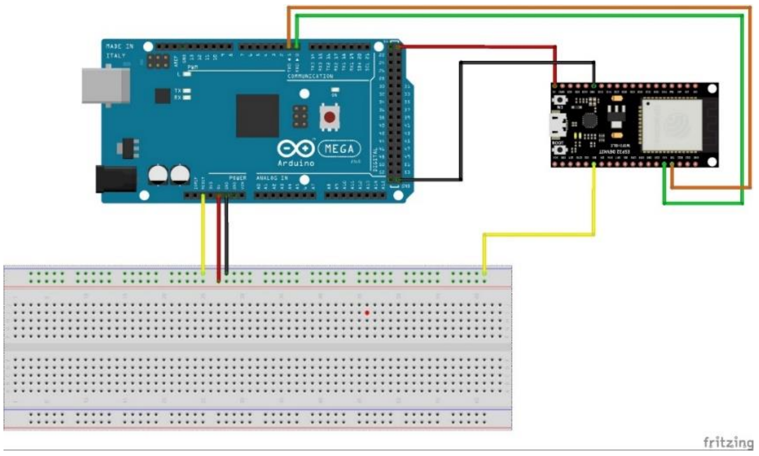


Figure 26: ESP32 connection in the project

5. CONSTRAINTS

Constraint 1: The robot should be controlled remotely in an open area having a radius of minimum 5 meters.

> We used a Bluetooth module (HC-06) that communicates between robot and mobile device from 10 m distance. In this way, we can control our robot from 10 meters distance.

Constraint 2 and 3: The robot should measure humidity and temperature. Additionally it should transfer this data remotely.

> We used a sensor (DHT) to achieve this constraint. This sensor has ability to measure temperature and humidity. Also by using bluetooth module we can see the data in the mobile application every 5 second.

Constraint 4: The robot should take the picture of the plants and transfer the picture data remotely.

> ESP32 Cam satisfies this constraint in our project. ESP32 module has a camera and it has ability to make a live stream by using Wi-Fi connection. We took module's IP address, and we specify this address in mobile application. When we open the link that contains the IP address we can see display the camera view.

Constraint 5: The designed product should efficiently operate in farm conditions.

> Farm conditions can be very hard, mud, rocks, grass etc. We want to design a robot that has high torque on its wheels, so we used two dc motors. In this way, current (which drives the motors) is not divided. Torque of the motors will be higher

(torque=constant of the motor*flux*armature current). If we use 4 motor we must connect 2 motor at the same output of the motor driver and current will be divided to two (motor driver has 2 couples of output pins). Separately, we used a servo motor to provide right and left side motion. In this way our battery will last longer. In addition, we used thick plastic material for platform to increase strength of the robot.

6. TASK DISTRIBUTION AMONG STUDENTS

We distributed the tasks in such a way that all members will bee in charge of one part. Also all members stayed in touch and work together.

Mücahit SAYİN dealt with mobile application. He searched to learn how mobile application can be designed with MIT App Inventor. He designed mobile application to satisfy motion control of the robot by a joystick and to display temperature and humidity data in mobile device's screen. Also he achieved the live broadasting on mobile phone part by coding on MIT app inventor.

Burak SEVİK worked the part of the project about live streaming with ESP32 module. He searched the Arduino library for the module and searched the code . He understood the working principle of module and worked to implement his knowledge over farm robot project.

Mehmet ERCİN dealt with motion control of the robot. He searched to learn operating principle and implemented the Arduino codes of the dc motors and servo motor. Also, he learned about joystick configuration. After that, he achieved his task and completed the Arduino codes for motion control and adapted for the mobile application.

Mehmet Zana SÖNER worked for measuring temperature and humidity by using DHT11 sensor. He made some researches about temperature and humidity sensors. He determined which sensor we will use. He searched about DHT11 sensor, and he worked for project with his information.

7. CONCLUSION

The basic idea of making Bluetooth controlled farm robot project is to help farmers with sensors, camera and ability of motion. Robot controls the condition of the area by sensor and camera and farmers will see this data in mobile application (which we have designed). If plant has a problem, farmer will figure out it thanks to the data. For example; if temperature is low, farmer will operate heating systems and make it to its optimum value. Also, farmer can check and control the situations of the plants by using the live streaming feature. By using this robot farmers will gain time and ease. This robot also increase efficiency in agriculture because farmers can recognize the diseases of the grains very early thanks to sensors and camera. If requires a intervention, farmers will interfere in early phase of the disease. In this manner farmers can save the plants and products.

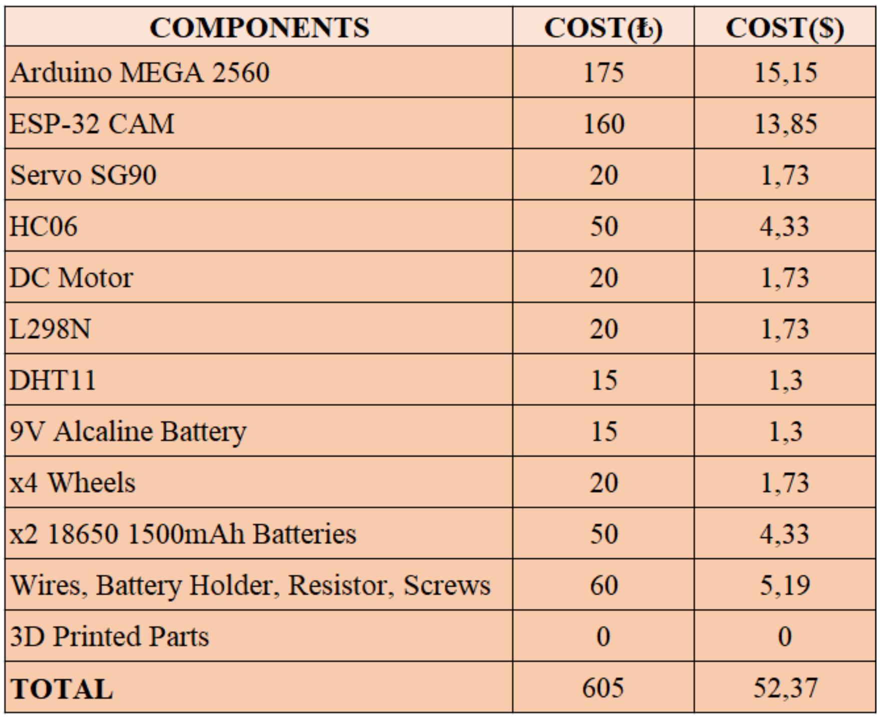
By achieving this project, we hope that it provides a contribution to the ever-developing modern agriculture, and an extra time and knowledge to farmers.

REFERENCES

1. http://gidabeslenme.org/modern-tarim/
2. https://en.wikipedia.org/wiki/Agriculture
3. https://www.dunyahalleri.com/farmbot-ile-tarimda-otomasyon-devri/
4. https://www.robotistan.com/6-v-250-rpm-motor-yellow-motor
5. https://community.appinventor.mit.edu/t/app-hangs-everytime-the-timer-ticks/46414
6. https://community.appinventor.mit.edu/t/virtual-joystick-coordinate-swap-problem-arduino-rc-car/48197
7. https://www.arduino.cc/reference/en/language/functions/communication/serial/
8. https://www.arduino.cc/reference/en/language/functions/time/millis/
9. https://www.arduino.cc/reference/en/language/functions/math/map/
10. https://create.arduino.cc/projecthub/pibots555/how-to-connect-dht11-sensor-with-arduino-uno-f4d239
11. https://create.arduino.cc/projecthub/sherbin/autonomous-assistant-agricultural-bots-1c85b2
12. https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/
13. https://forum.arduino.cc/t/read-char-over-serial-solved/153697
14. https://www.tinkercad.com/things/6FEj95LTFpp-copy-of-cheap-rc-car-servo-steering
15. https://community.kodular.io/t/how-to-create-a-joystick-with-the-canvas-component/58129
16. https://community.kodular.io/t/phase-animations-made-easy/43513
17. https://www.youtube.com/c/mcwhorpj
18. https://youtu.be/RM0G7q-G6bo
19. https://youtu.be/HBQIH98wmHs
20. https://youtu.be/RM0G7q-G6bo
21. https://youtu.be/BT4YihNXiYw
22. https://www.quora.com/What-value-and-type-of-battery-should-I-use-for-my-RC-car-project
23. https://components101.com/wireless/hc-06-bluetooth-module-pinout-datasheet
24. https://learn.adafruit.com/dht
25. https://store.arduino.cc/products/arduino-mega-2560-rev3
26. https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/
27. https://create.arduino.cc/projecthub/noah_arduino/using-esp32-cam-with-arduino-b4f12c
28. https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf

COST ANALYSIS AND GANTT CHART

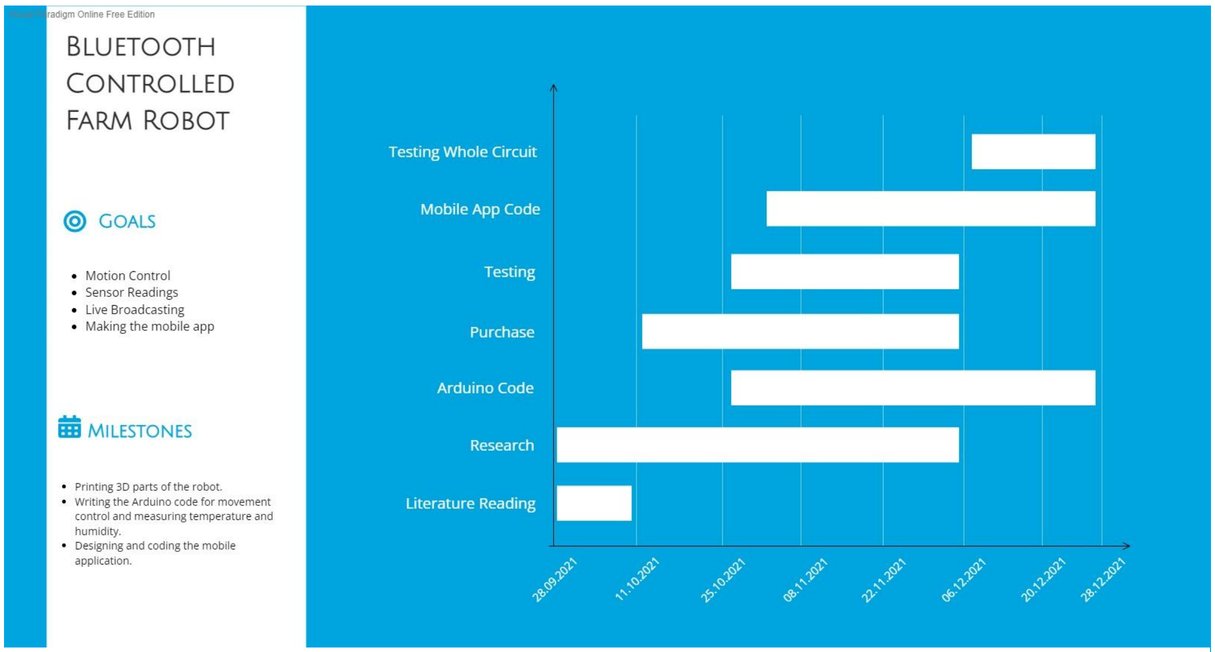
Cost Analysis



COMPONENTS	COST(₺)	COST($)
Arduino MEGA 2560	175	15,15
ESP-32 CAM	160	13,85
Servo SG90	20	1,73
HC06	50	4,33
DC Motor	20	1,73
L298N	20	1,73
DHT11	15	1,3
9V Alcaline Battery	15	1,3
x4 Wheels	20	1,73
x2 18650 1500mAh Batteries	50	4,33
Wires, Battery Holder, Resistor, Screws	60	5,19
3D Printed Parts	0	0
TOTAL	605	52,37

Figure 27: Cost analysis

Gantt Chart



BLUETOOTH CONTROLLED FARM ROBOT

GOALS

- Motion Control
- Sensor Readings
- Live Broadcasting
- Making the mobile app

MILESTONES

- Printing 3D parts of the robot.
- Writing the Arduino code for movement control and measuring temperature and humidity.
- Designing and coding the mobile application.

Figure 28: Gantt chart