

2022

# Arduino Car

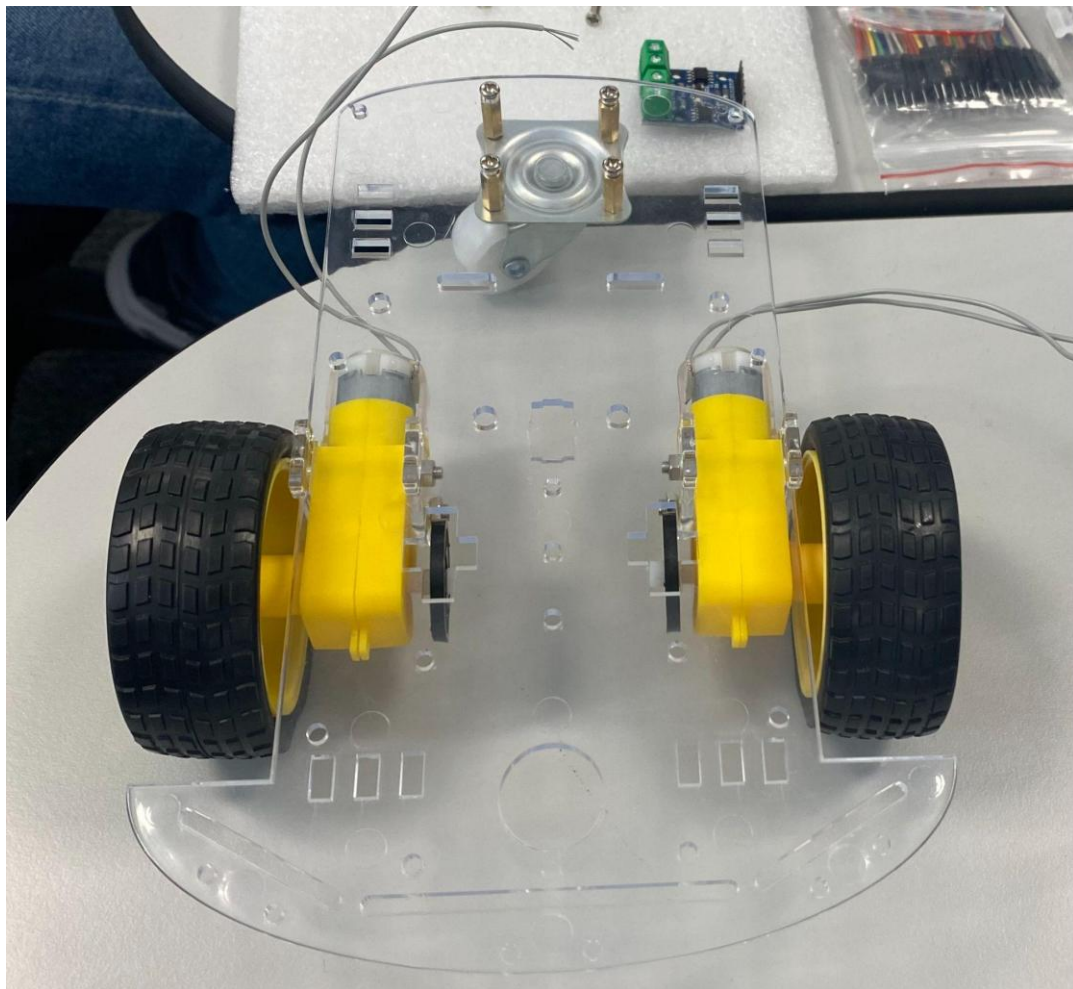
In this lab session, start building our robot car and write a simple program to control the motors.

Task 1. Watch the YouTube video titled “How to assemble the 2WD Smart Robot Car Chassis Kit for Arduino” using the link below:  
<https://www.youtube.com/watch?v=H78t6dnSoG0>.

We watched the video from youtube and made ourselves familiar with the sequence of assembly of the smart robot car.

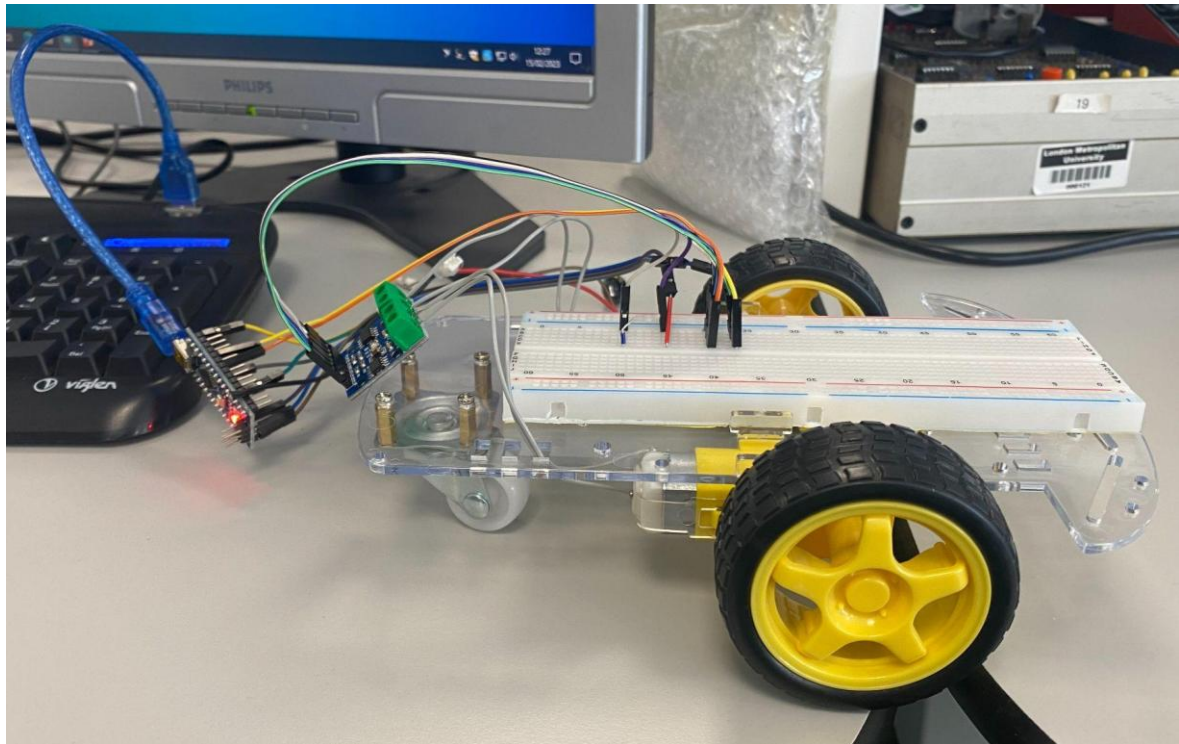
Task2. Get power cables soldered to both of your motors

Task3. Mount the DC motors, drive wheels and castor wheel to chassis as explained in task 1 above.



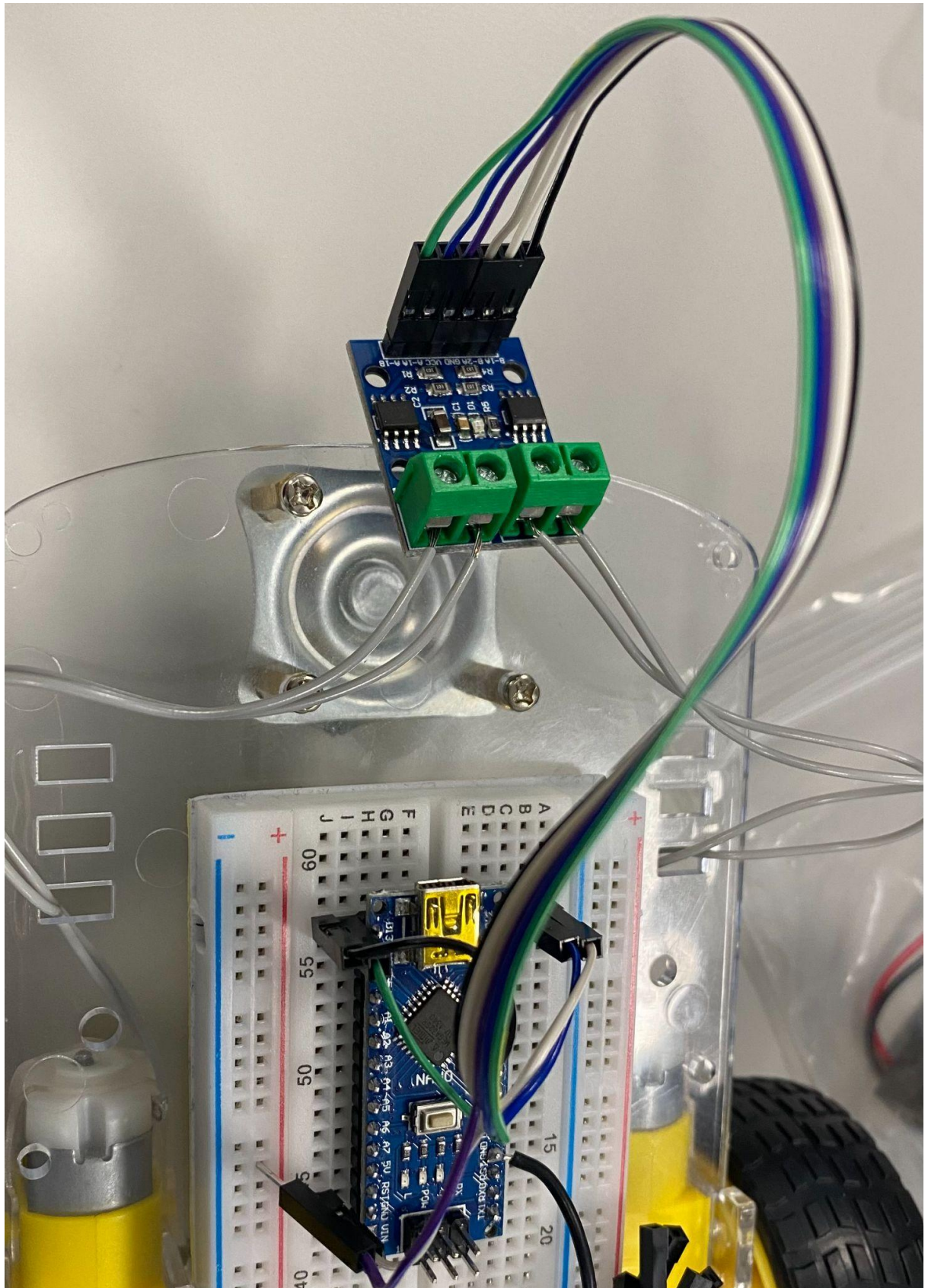
We have soldered the cables to both motors and installed them on the body car. We also fitted the wheels.

Task4. Connect the H-bridge driver to Arduino and your motor power cables as shown in the diagram below:



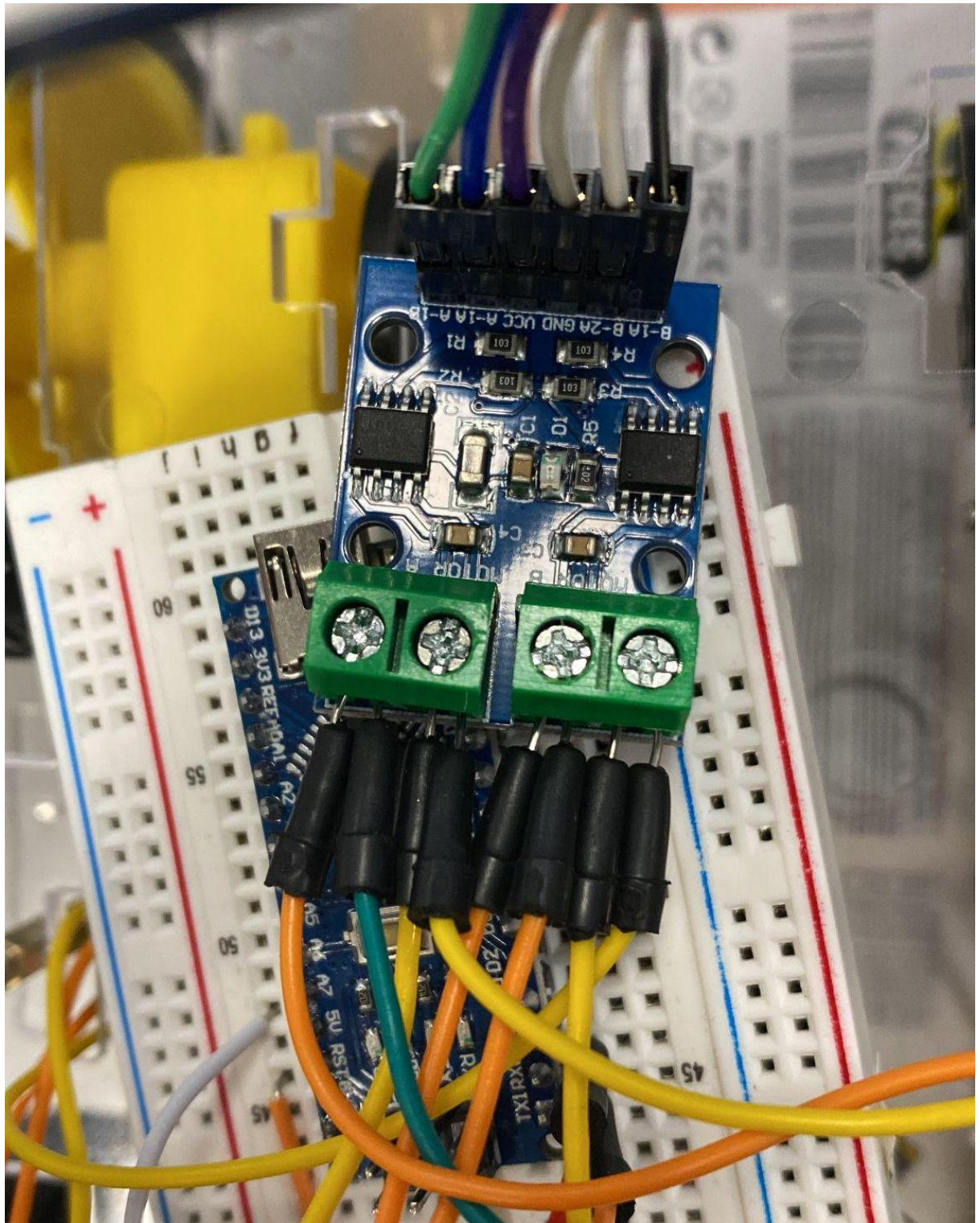
Here you can see our motors connected to the H-bridge which is connected to the Arduino Nano by cables.





In this picture you can see our second attempt where we connect our Arduino directly to the breadboard to avoid spaghetti mix.





This picture is from our second car which has 4 motors connected to the H-bridge.

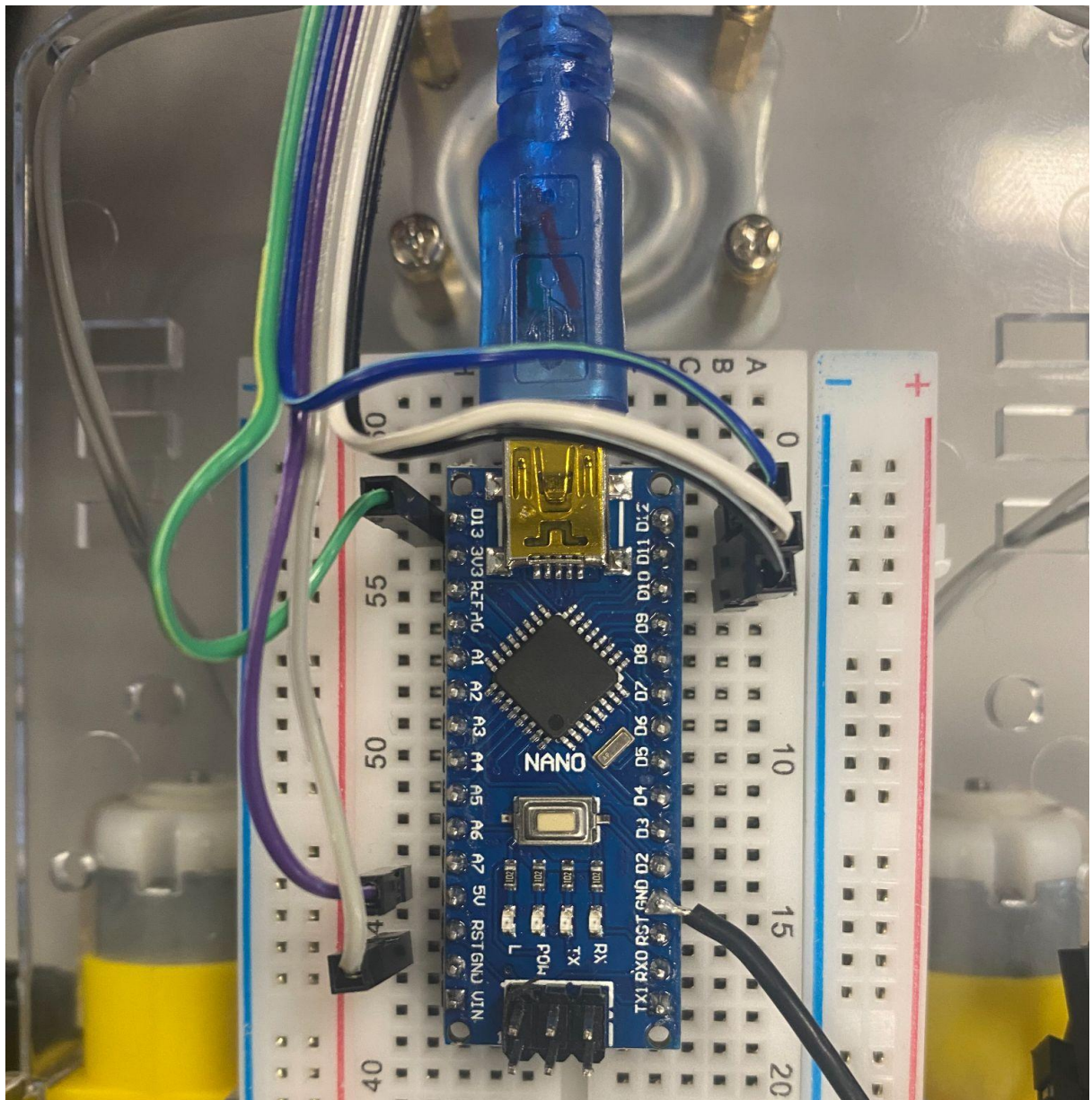


## 22-Feb-2023 Sensors and Actuators: Motor Control Functions and Bluetooth Setup / Robot Car Motor Control (Week 16)

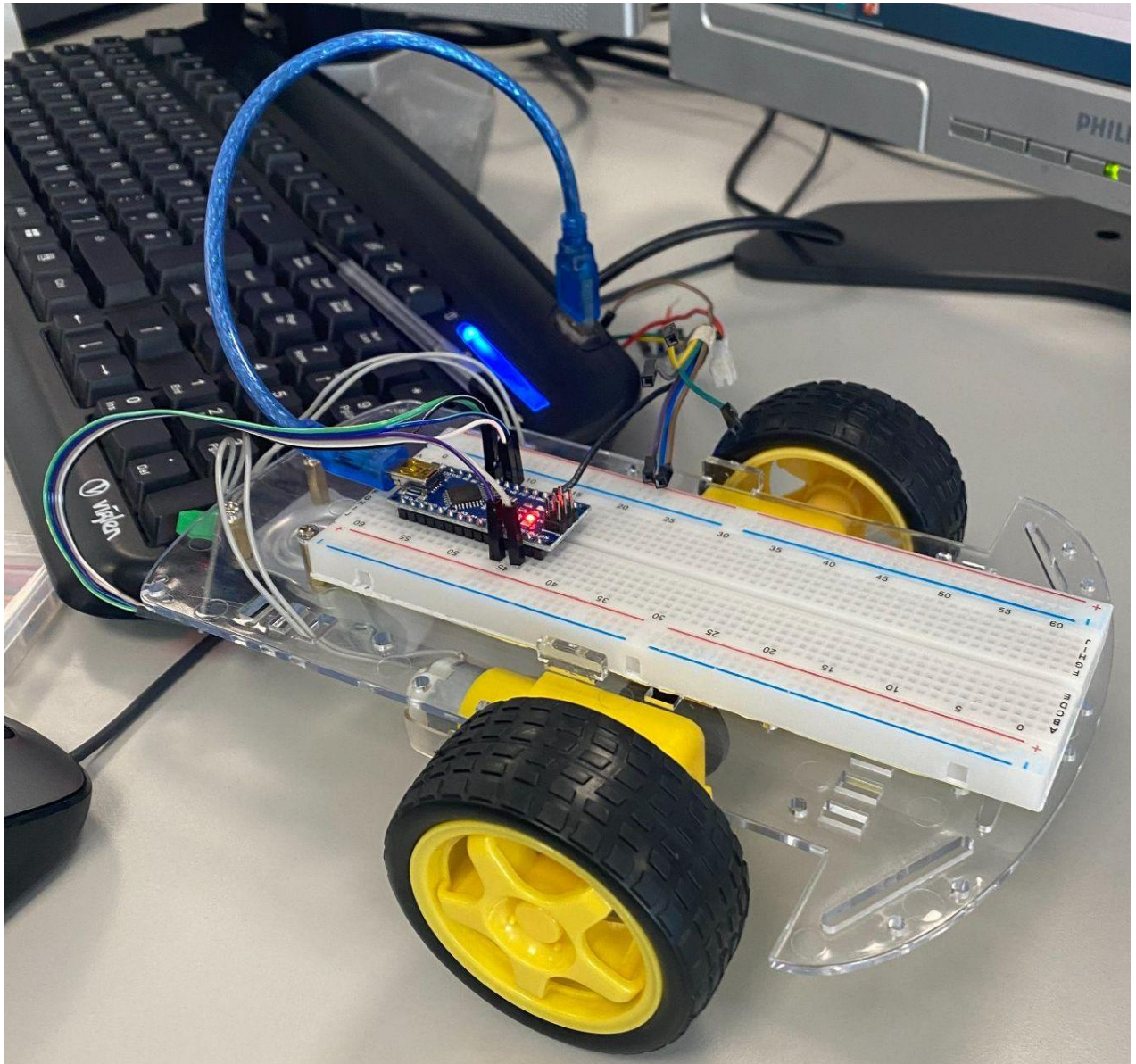
Robot Car Motor Control - 2 In this lab session, we will write C functions in order to control the robot car. Please complete the previous workshop titled “Robot Car Construction & Motor Control” if you have not already done so. For these workshops, you may find it easier to power your robot car using a 5V rechargeable power pack.

Tasks: 1.

Connect both of the car motors to the H-Bridge driver. Connect H-Bridge driver power and control pins to the Arduino board, making note of the control and Arduino pins used in your logbook.



We connected both motors to the H-Bridge driver which are connected to the Arduino board, and they are working perfectly. Here we used pins 10 to 13 on Arduino Nano board to send a signal to H-bridge.

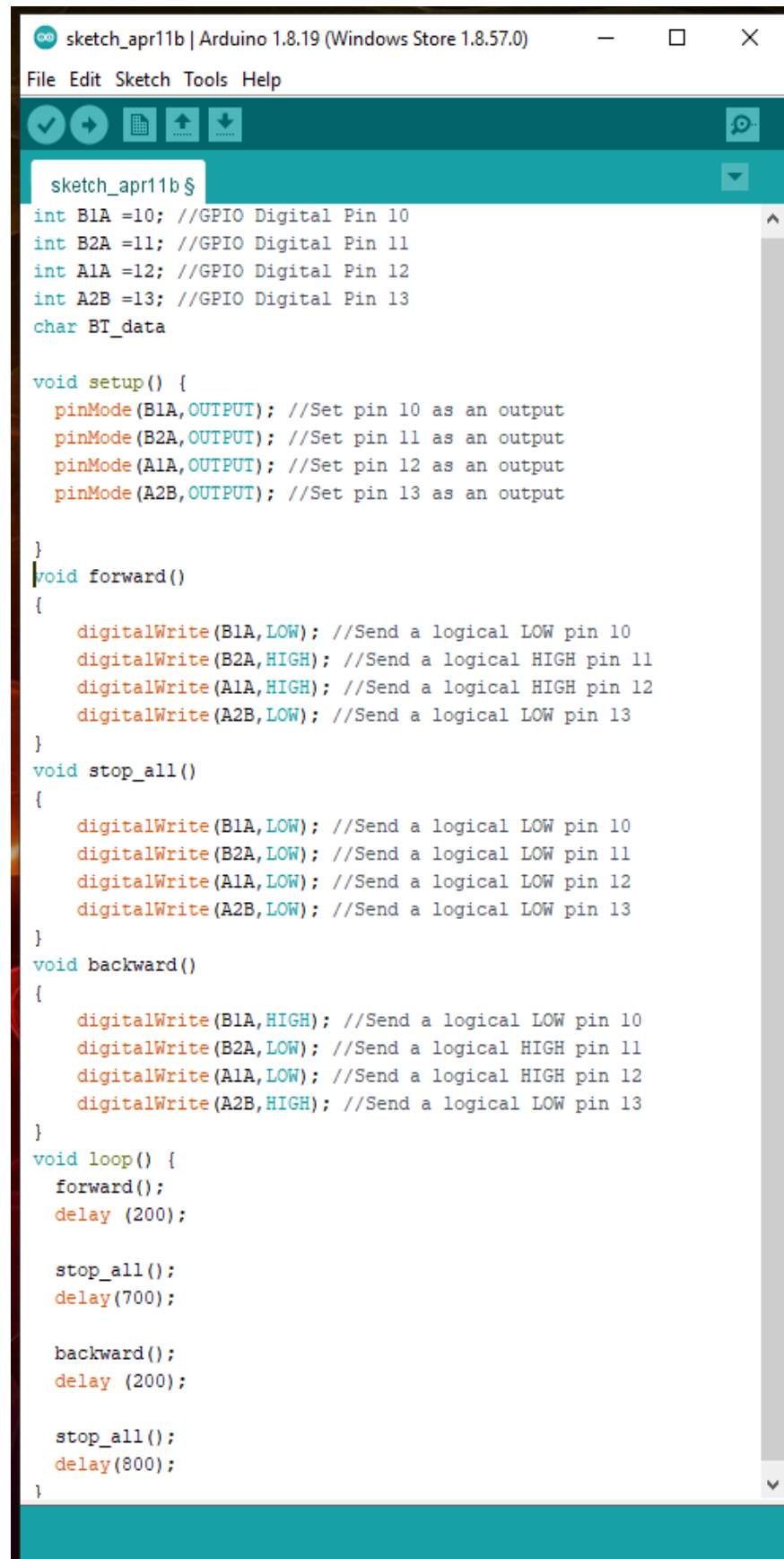


In this case we have changed the pins from 10,11,12,13 to 4,5,6,7 for more comfort.



## Tasks: 2.

Write a program that will move the car forward for 1 second and wait for half a second. Then move the car backwards for 1 second and wait for half a second. Then repeat the entire cycle.

The image shows a screenshot of the Arduino IDE interface. The title bar at the top reads "sketch\_apr11b | Arduino 1.8.19 (Windows Store 1.8.57.0)". Below the title bar is a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Under the "Sketch" menu, there are icons for Verify, Upload, and Serial Monitor. The main text area contains the following C++ code:

```
sketch_apr11b $
int B1A =10; //GPIO Digital Pin 10
int B2A =11; //GPIO Digital Pin 11
int A1A =12; //GPIO Digital Pin 12
int A2B =13; //GPIO Digital Pin 13
char BT_data

void setup() {
  pinMode(B1A,OUTPUT); //Set pin 10 as an output
  pinMode(B2A,OUTPUT); //Set pin 11 as an output
  pinMode(A1A,OUTPUT); //Set pin 12 as an output
  pinMode(A2B,OUTPUT); //Set pin 13 as an output
}
void forward()
{
  digitalWrite(B1A,LOW); //Send a logical LOW pin 10
  digitalWrite(B2A,HIGH); //Send a logical HIGH pin 11
  digitalWrite(A1A,HIGH); //Send a logical HIGH pin 12
  digitalWrite(A2B,LOW); //Send a logical LOW pin 13
}
void stop_all()
{
  digitalWrite(B1A,LOW); //Send a logical LOW pin 10
  digitalWrite(B2A,LOW); //Send a logical LOW pin 11
  digitalWrite(A1A,LOW); //Send a logical LOW pin 12
  digitalWrite(A2B,LOW); //Send a logical LOW pin 13
}
void backward()
{
  digitalWrite(B1A,HIGH); //Send a logical LOW pin 10
  digitalWrite(B2A,LOW); //Send a logical HIGH pin 11
  digitalWrite(A1A,LOW); //Send a logical HIGH pin 12
  digitalWrite(A2B,HIGH); //Send a logical LOW pin 13
}
void loop() {
  forward();
  delay (200);

  stop_all();
  delay(700);

  backward();
  delay (200);

  stop_all();
  delay(800);
}
```



We wrote a code which contains three functions that make the car go forward, stop the car and go backward. In the loop section we call this function so it will do it as long as we power the car.

Tasks: 3.

Write two C functions that will rotate the car about 90 degrees to the left and 90 degrees to the right and test that your C functions are working as expected.

```
void left()
{
    digitalWrite(B1A,HIGH); //Send a logical LOW pin 10
    digitalWrite(B2A,LOW); //Send a logical HIGH pin 11
    digitalWrite(A1A,HIGH); //Send a logical HIGH pin 12
    digitalWrite(A2B,LOW); //Send a logical LOW pin 13
}

void right()
{
    digitalWrite(B1A,LOW); //Send a logical LOW pin 10
    digitalWrite(B2A,HIGH); //Send a logical HIGH pin 11
    digitalWrite(A1A,LOW); //Send a logical HIGH pin 12
    digitalWrite(A2B,HIGH); //Send a logical LOW pin 13
}

void loop()
{
    left();
    delay(670);

    stop_all();
    delay(1000);

    right();
    delay(670);

    stop_all();
    delay(1000);
}
```

We wrote two functions which will turn the car left for 90 degrees and right to 90 degrees.

Tasks: 4.

Using the functions written for task 3, write a program such that the car goes forward for 1 second, rotates 180 degrees and moves forward again for the same duration. It should then rotate 180 degrees in the opposite direction facing the original direction.

```
void loop()
{
    forward();
    delay (1000);

    stop_all();
    delay(500);

    left();
    delay(1380);

    stop_all();
    delay(500);

    forward();
    delay (1000);

    stop_all();
    delay(500);
    |
    right();
    delay(1380);

    stop_all();
    delay(500);

}
```

In this situation car is moving forward for one second turning left for 180 degrees after goes forward for one second and turn right for 180 degrees



Tasks: 5.

Write a program to move the car such that its path traces a square.

```
void loop()
{
  forward();
  delay (1000);

  stop_all();
  delay(500);

  left();
  delay(670);

  stop_all();
  delay(500);
}
```

The car will go forward for one second, stop for half second and turn left for 90 degrees and will repeat it till we power it, which will result a square.

```
void loop()
{
  backward();
  delay (1000);

  stop_all();
  delay(500);

  right();
  delay(670);

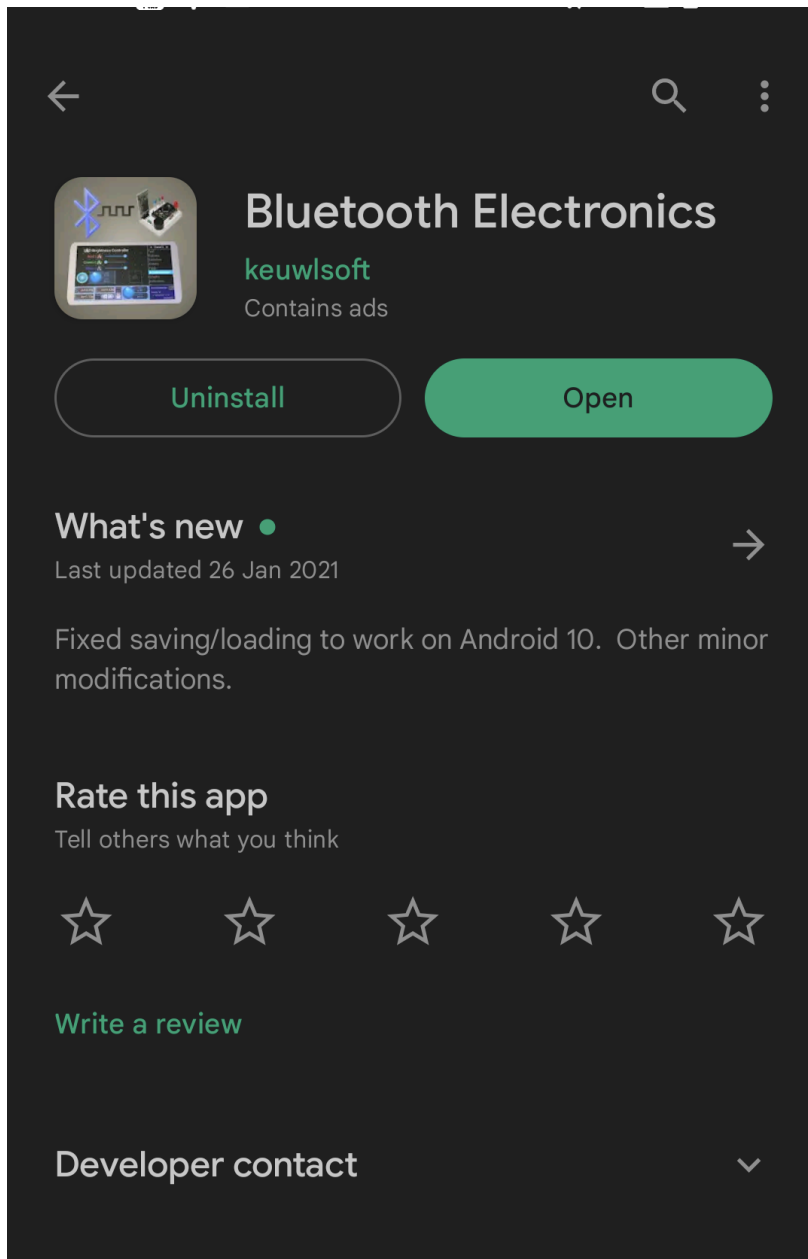
  stop_all();
  delay(500);
}
```

Here the car will do the same thing but in the opposite direction. It will go backward for one second, stop for half a second and turn right for 90 degrees and will repeat it till we power it.

In this lab session, we will continue using Arduino IDE (Integrated Development Environment) and will be interfacing a Bluetooth transceiver module through a serial port. Please make sure all your programs are saved on your USB with different names so that you can easily access anyone of them.

Tasks: 1.

Download and install “Bluetooth Electronics” app from Google Play Store on an Android device.

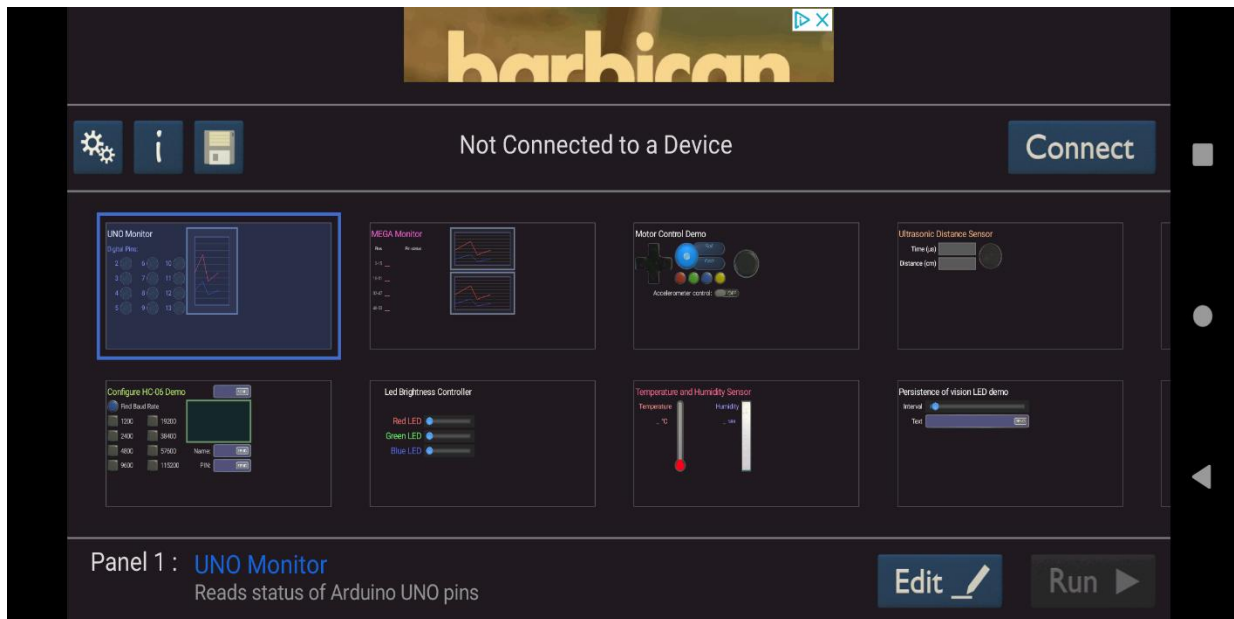


Found the right to connect to Arduino nano via Bluetooth and installed it.

Tasks: 2.

Examine the devices available in the app, experiment with devices by editing a panel, check device properties.

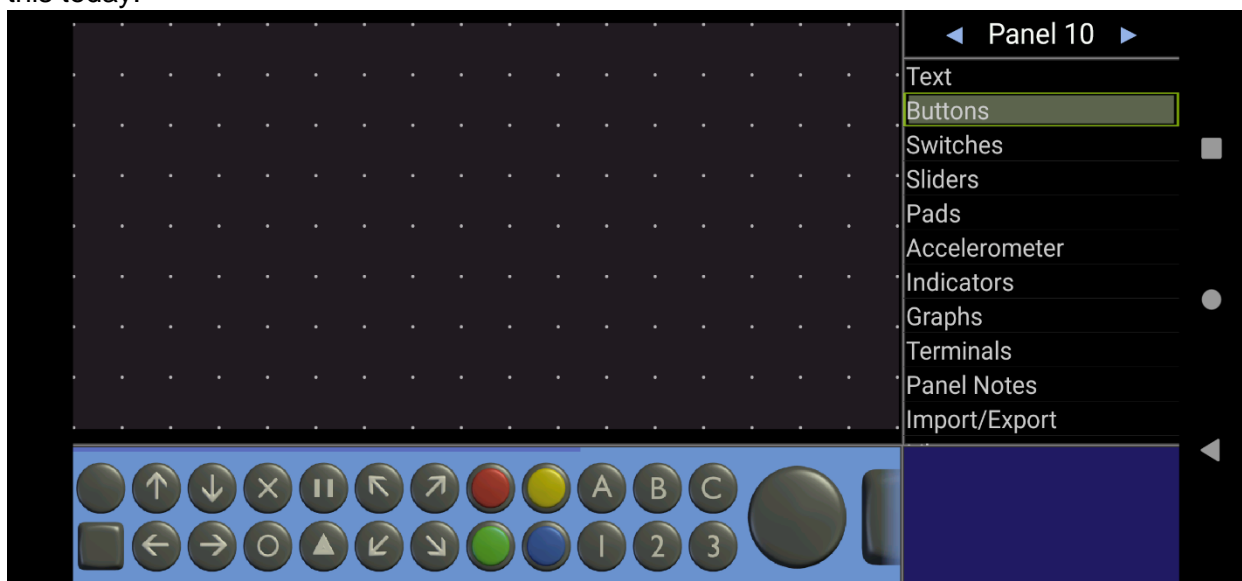




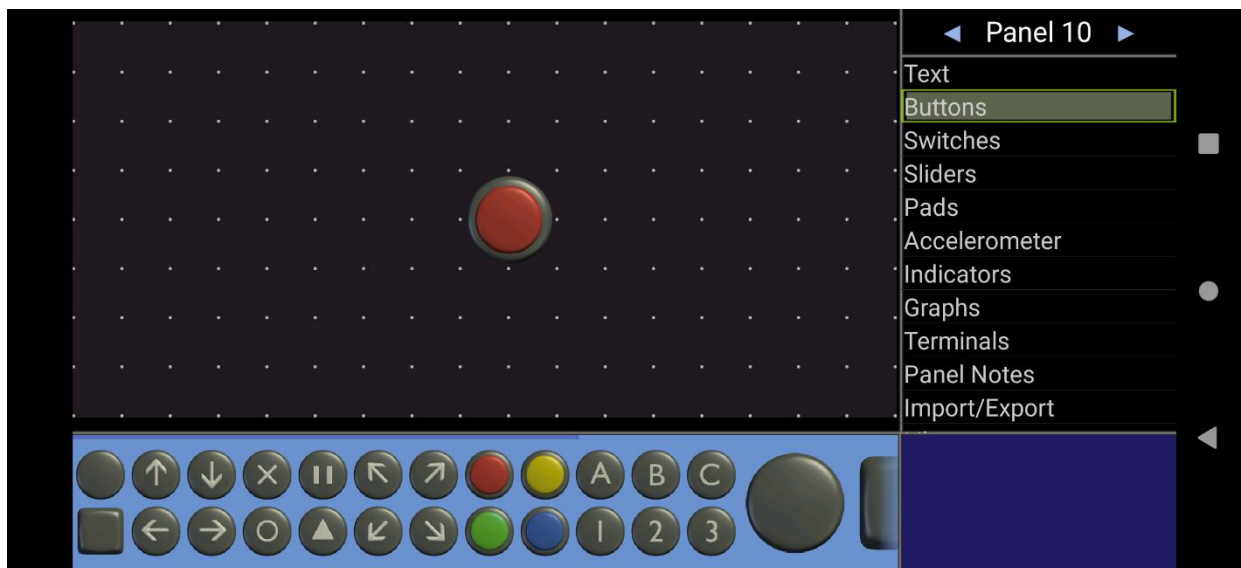
I examined the app and make myself familiar with it.

Tasks: 3.

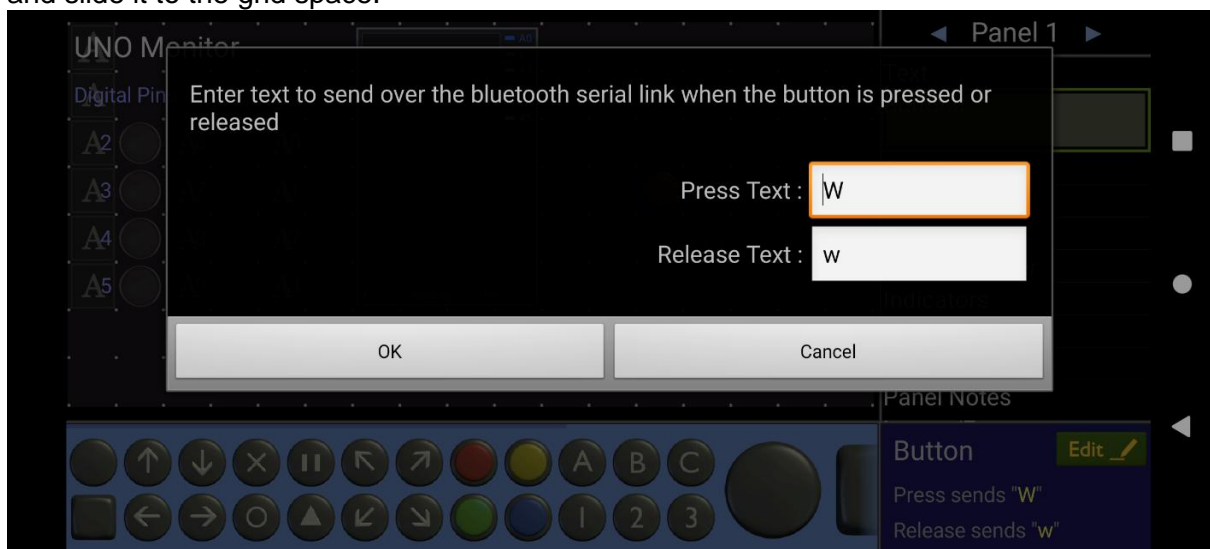
Prepare a panel with a single “Button” on the user screen. The user screen should include the module code and module title as well as names of the group members who are working on this today.



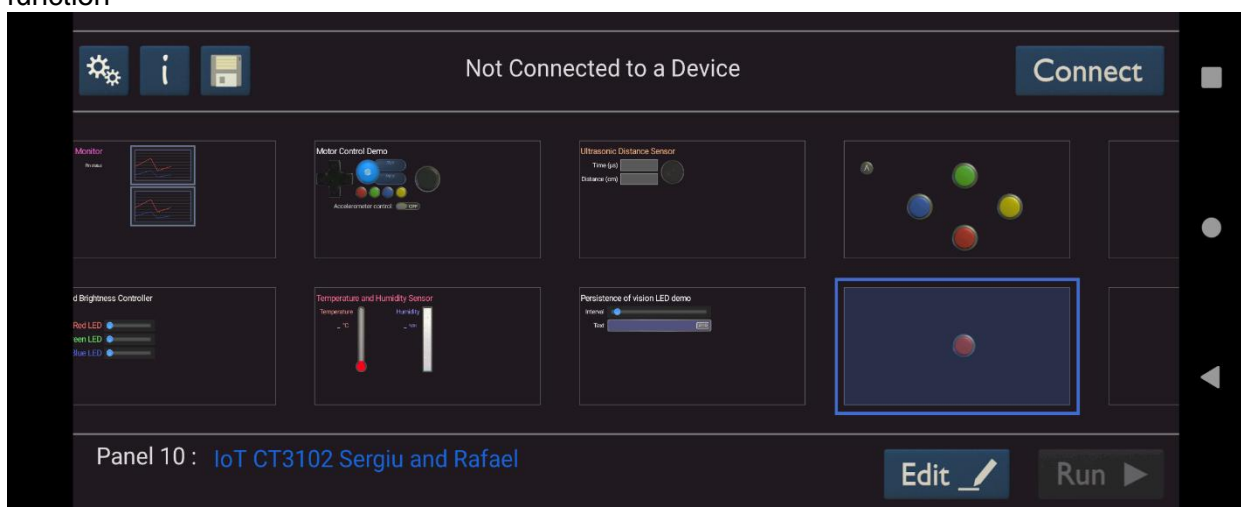
We chose a new sketch where we will build our buttons to control the Arduino Nano. On the right side we have a list of features.



From the list on the right side of the screen we select Buttons section and select the button and slide it to the grid space.



After we placed the button in the bottom left corner of the screen will appear edit function of the button. Here we write the letter from our Arduino code which will activate an particular function

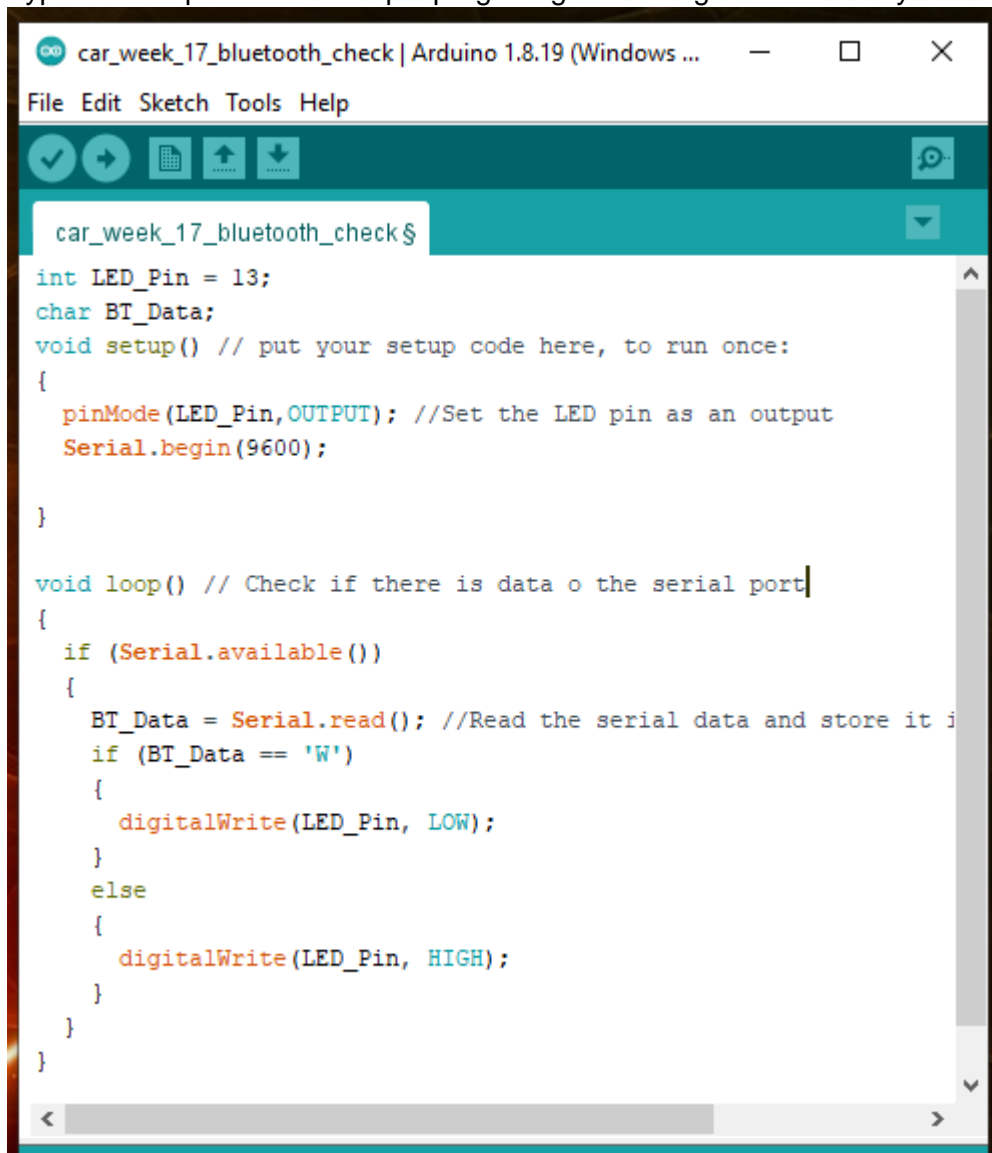


On this image you can see the name of the sketch



Tasks: 4.

Type in and upload the example program given during the lecture to your Arduino board.

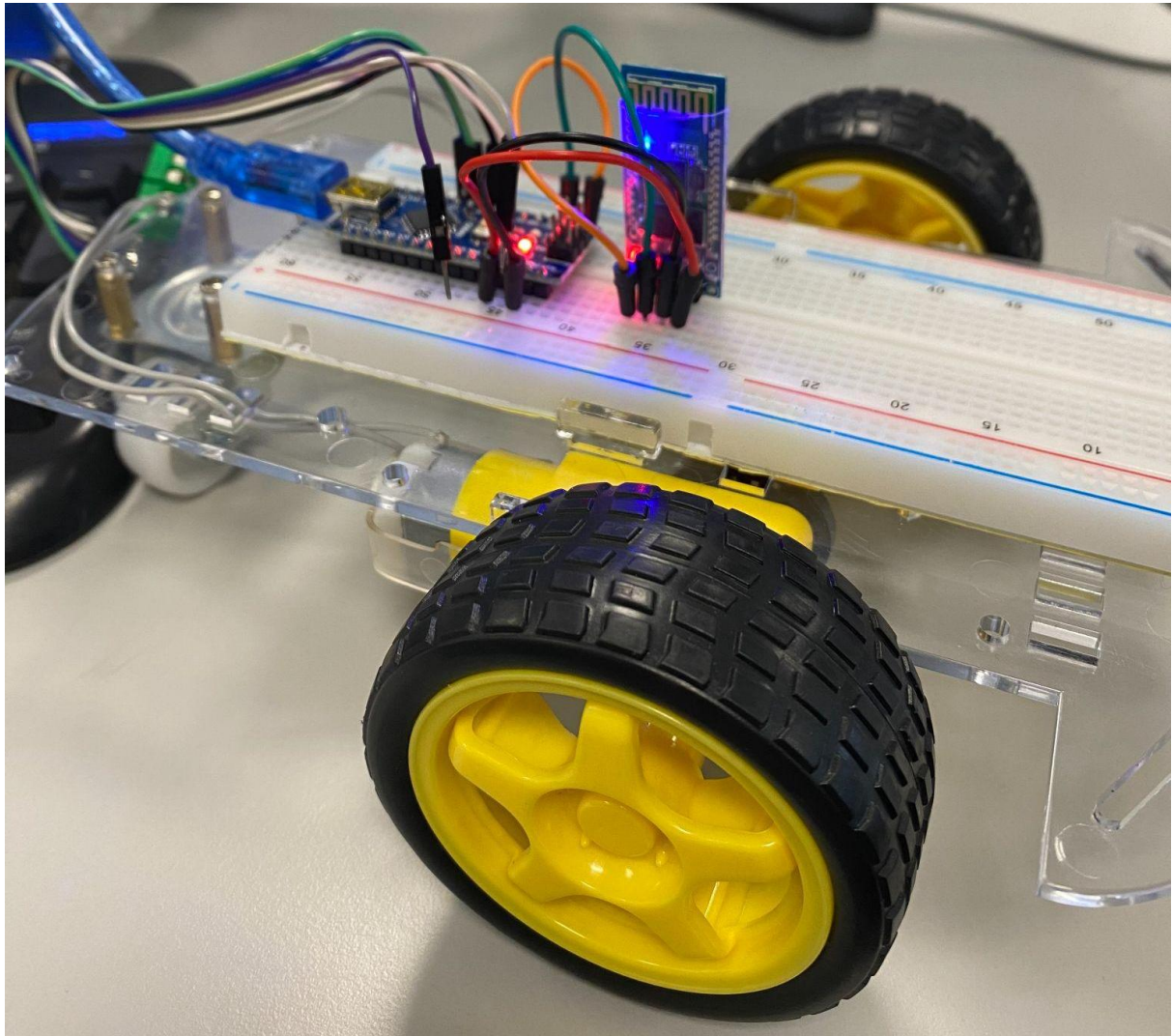
The image shows a screenshot of the Arduino IDE interface. The title bar reads "car\_week\_17\_bluetooth\_check | Arduino 1.8.19 (Windows ...". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening files, saving, uploading, and downloading. The main text area contains the following C++ code:

```
car_week_17_bluetooth_check$  
int LED_Pin = 13;  
char BT_Data;  
void setup() // put your setup code here, to run once:  
{  
  pinMode(LED_Pin,OUTPUT); //Set the LED pin as an output  
  Serial.begin(9600);  
}  
  
void loop() // Check if there is data o the serial port|  
{  
  if (Serial.available())  
  {  
    BT_Data = Serial.read(); //Read the serial data and store it i  
    if (BT_Data == 'W')  
    {  
      digitalWrite(LED_Pin, LOW);  
    }  
    else  
    {  
      digitalWrite(LED_Pin, HIGH);  
    }  
  }  
}
```

We wrote the code and send it to the Arduino nano to check if it is working.

Tasks: 5.

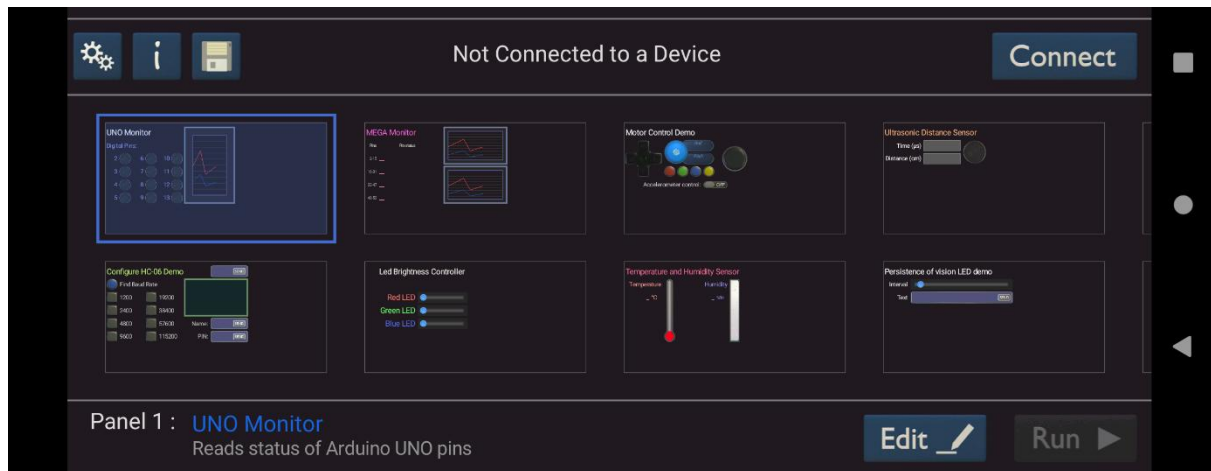
Disconnect your Arduino board from the USB cable and connect the Bluetooth module using four jumper cables as given during the lecture. Power the Arduino and the Bluetooth module from a +5V voltage source (USB port of your PC should also work fine).



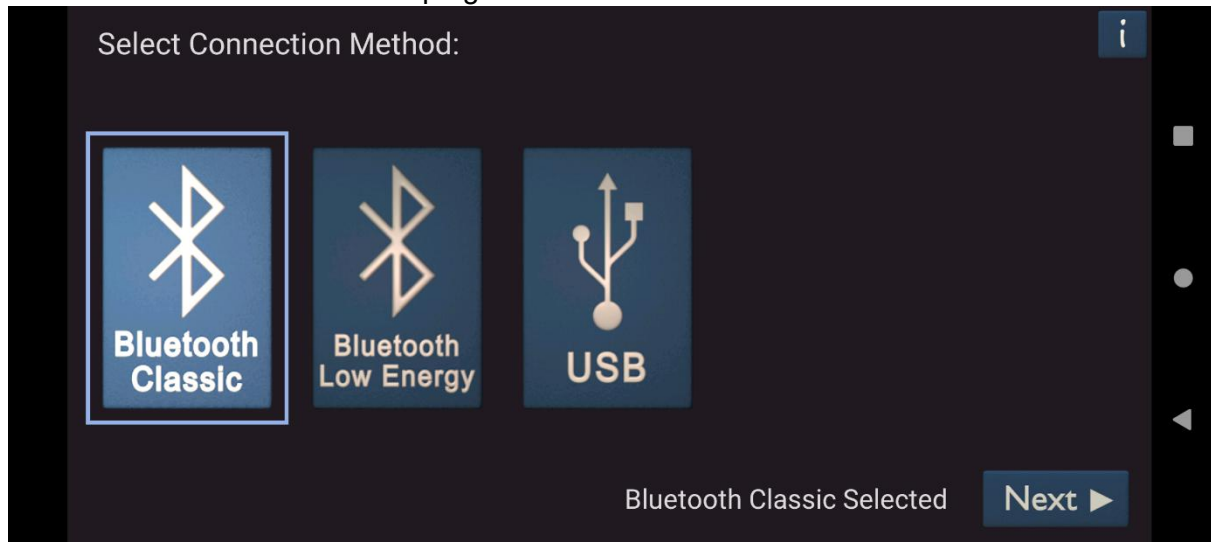
We have connected the Bluetooth module to the solderless breadboard which is already connected to the Arduino Nano. The Bluetooth module is connected with RX pin to TX pin on Arduino and TX pin from Bluetooth to RX on Arduino. It is powered by 5 Voltage from Arduino pin and grounded as well. By the flashing light on the Bluetooth module, we can understand that it is working.

Tasks: 6.

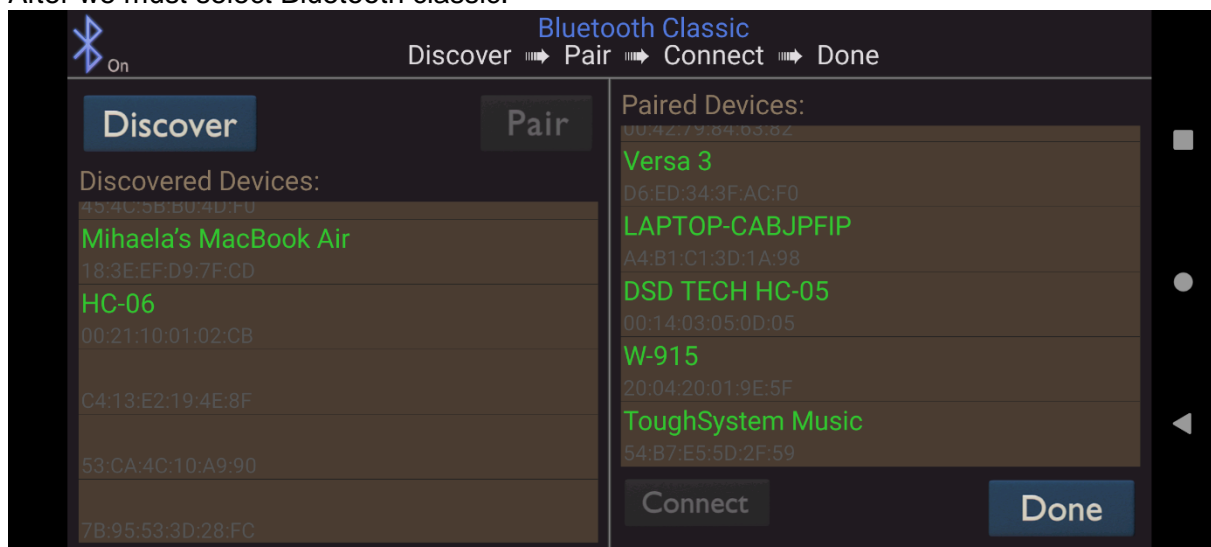
Establish a connection between your Android device and the Bluetooth module connected to the Arduino board.



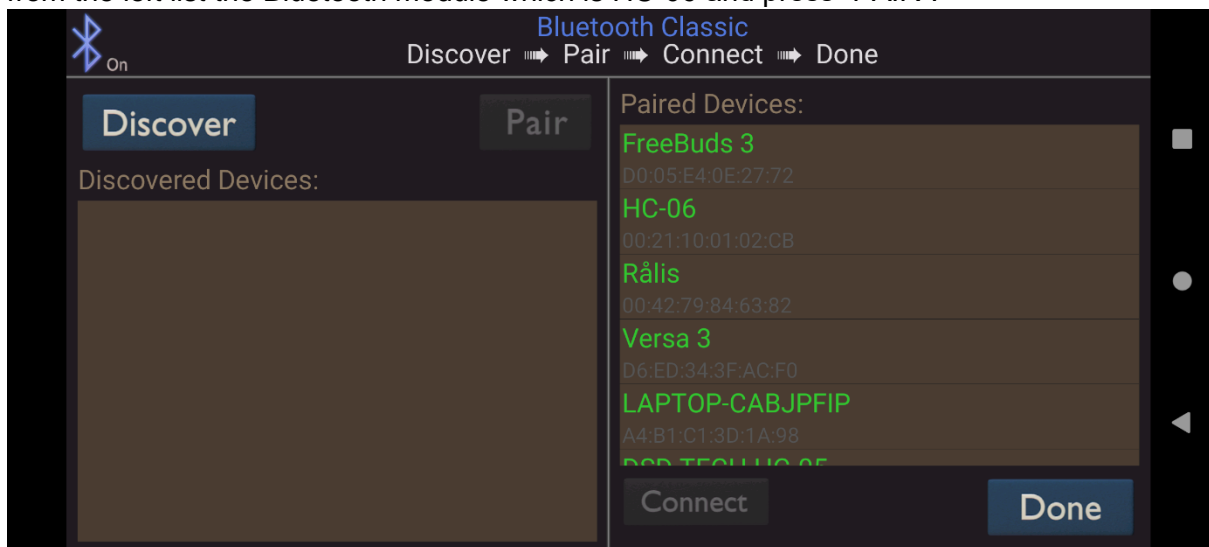
To establish connection between our device and the Bluetooth module we need to press "CONNECT" command in the top right corner of the screen.



After we must select Bluetooth classic.



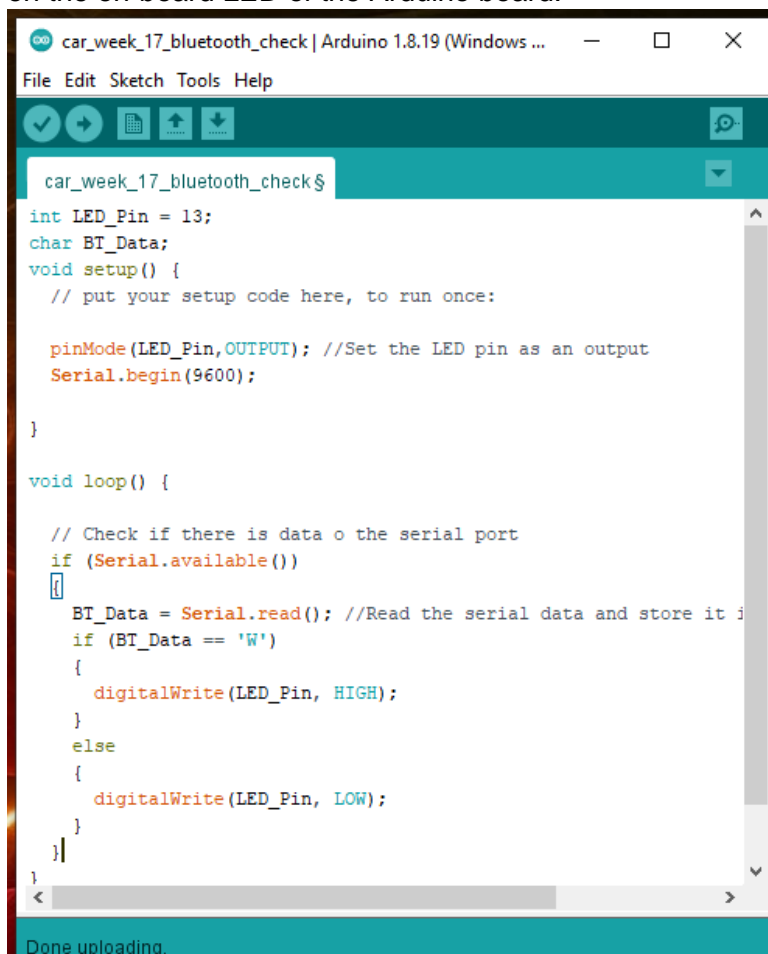
The next step we must select “DISCOVER” to detect our Bluetooth module, after we select from the left list the Bluetooth module which is HC-06 and press “PAIR”.



Now from the right list we must select our Bluetooth module which is “HC-06” and press “CONNECT” for the first time it will ask to insert the pin which is “0000” or “1234” by default after this we must wait a bit and press “DONE”.

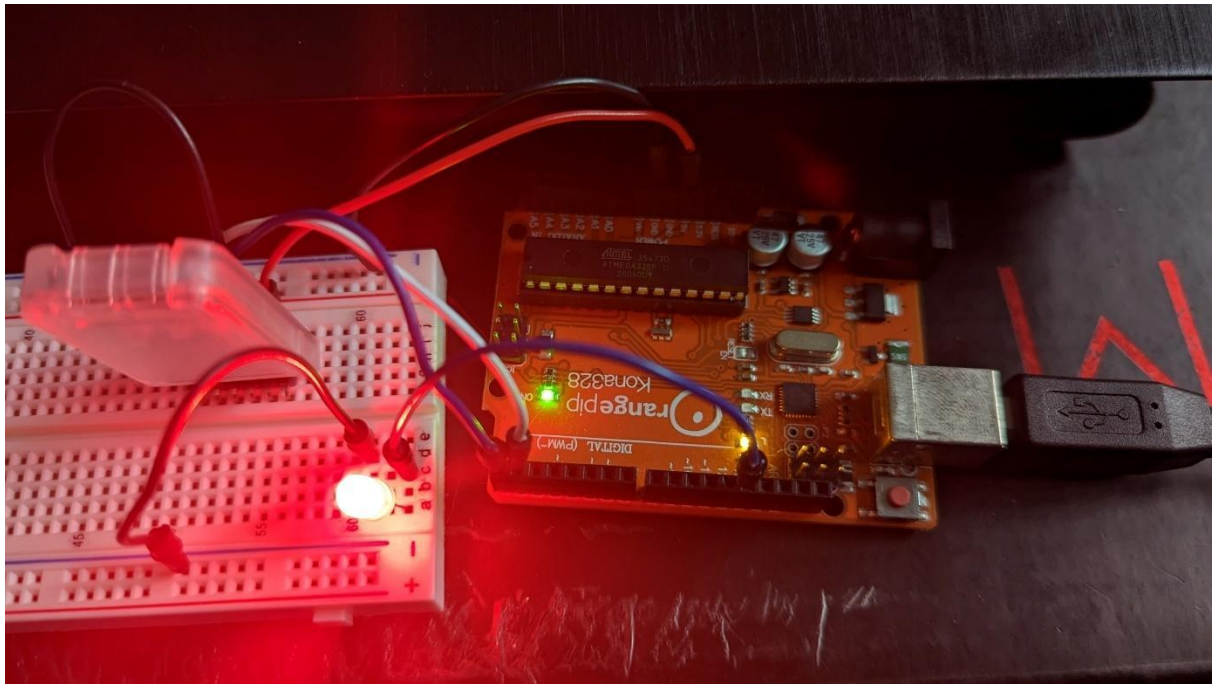
Tasks: 7.

Observe and record in your logbook the effect of pressing the Button on your Android device on the on-board LED of the Arduino board.



Above we wrote the code to check our Bluetooth module though the app.





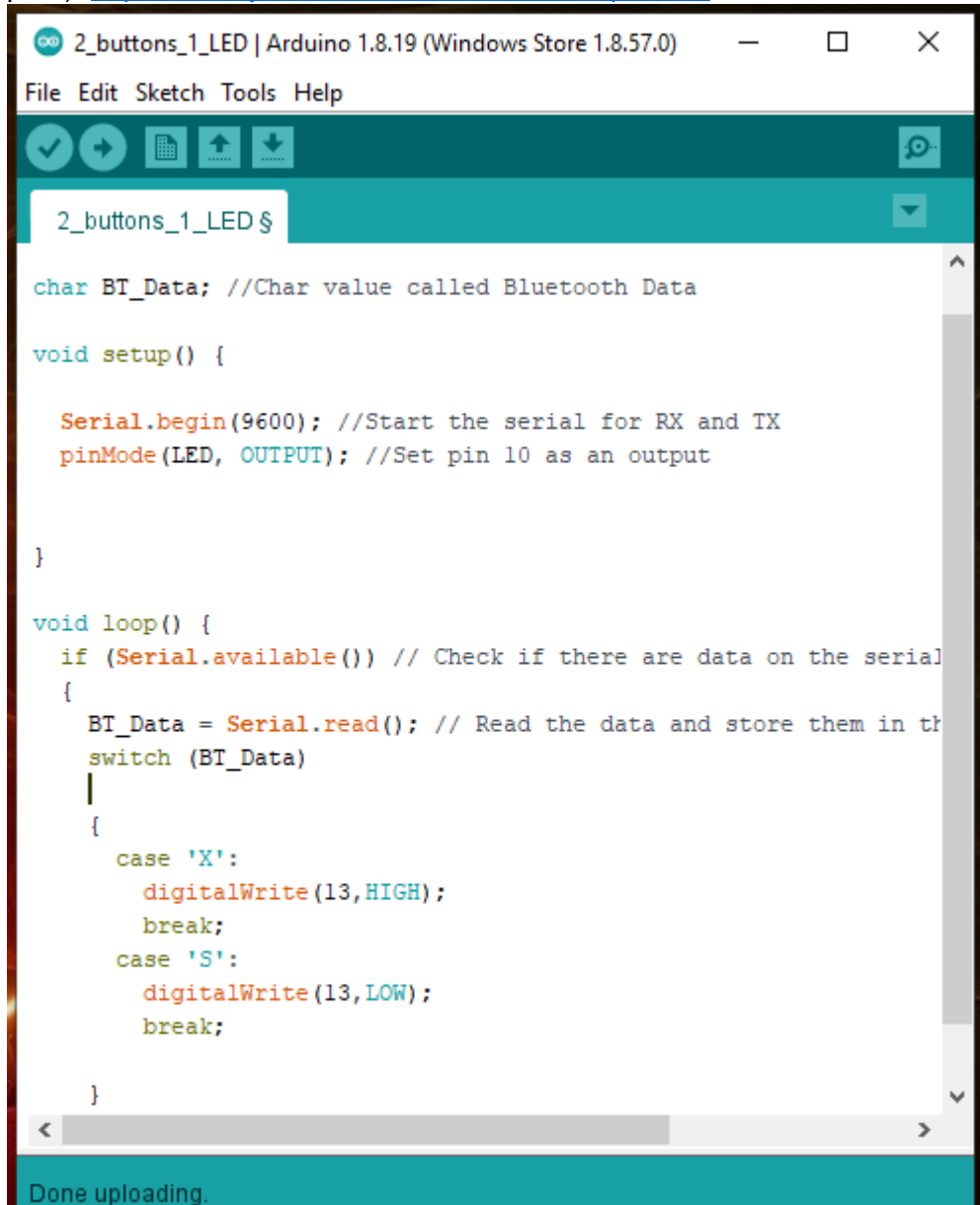
Here you can see the result when the button is pressed, everything is working right as expected. When we press the button the LED lighting when we release the button it stops lighting.

Tasks: 8.

Edit your Arduino program so that pressing the on screen "Button" once turns the LED on and pressing again turns it off. (You will need to disconnect Bluetooth RX and TX jumper cables to enable your new program to be uploaded and reconnected them once your program is uploaded.

Tasks: 9.

If time permits, connect two external LEDs to your Arduino board and switch these on or off using two buttons on the Bluetooth Electronics app screen. Further information about the use of the “Bluetooth Electronics” app is available at: <https://www.keuwl.com/electronics.html> You may also find the following YouTube video, helpful (even though the recording quality is rather poor): <https://www.youtube.com/watch?v=V6lRp30sfcs>

The image is a screenshot of the Arduino IDE interface. The title bar at the top reads '2\_buttons\_1\_LED | Arduino 1.8.19 (Windows Store 1.8.57.0)'. Below the title bar is a menu bar with 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. A toolbar with various icons is located below the menu bar. The main text area contains the following C++ code:

```
char BT_Data; //Char value called Bluetooth Data

void setup() {

    Serial.begin(9600); //Start the serial for RX and TX
    pinMode(LED, OUTPUT); //Set pin 10 as an output

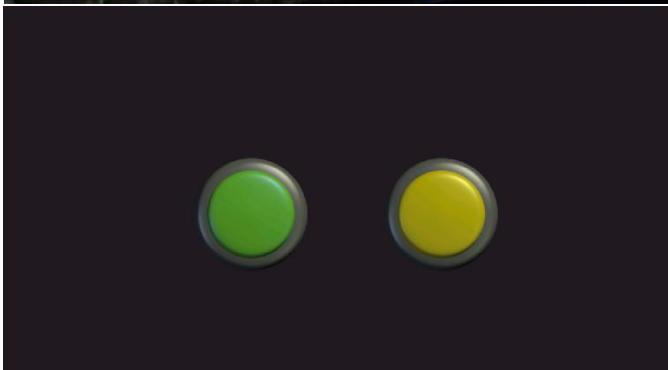
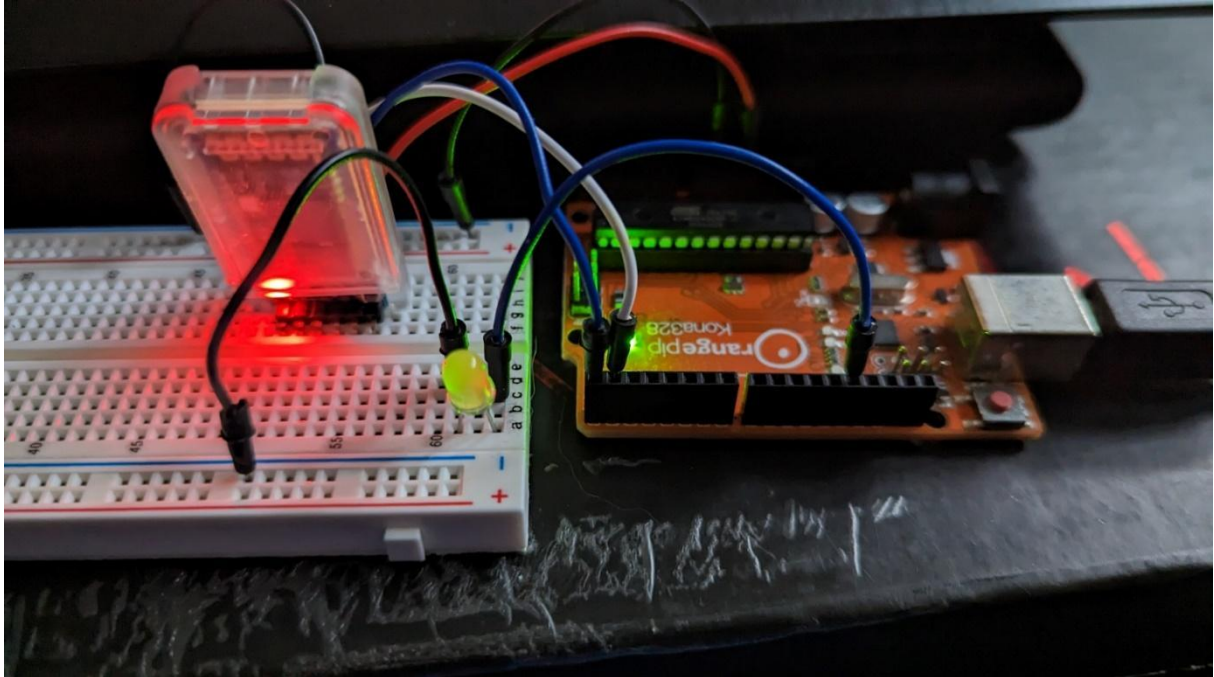
}

void loop() {
    if (Serial.available()) // Check if there are data on the serial
    {
        BT_Data = Serial.read(); // Read the data and store them in the variable
        switch (BT_Data)
        {
            case 'X':
                digitalWrite(LED, HIGH);
                break;
            case 'S':
                digitalWrite(LED, LOW);
                break;
        }
    }
}
```

At the bottom of the window, a status bar shows the text 'Done uploading.'.

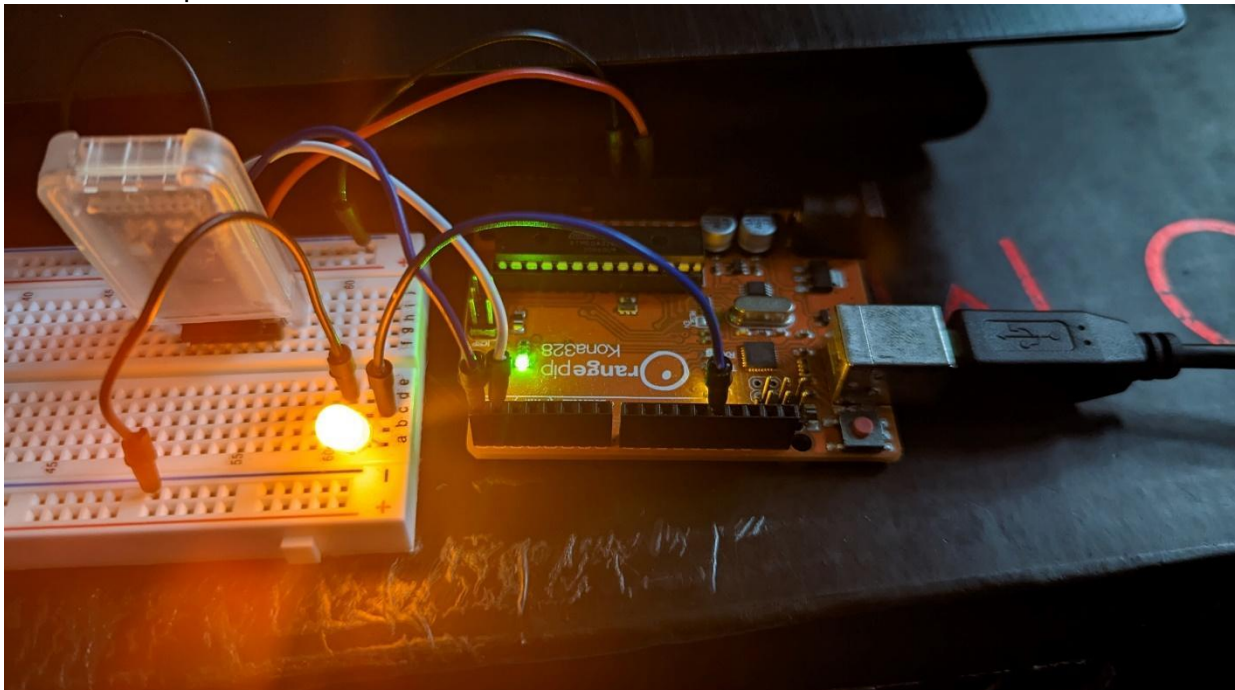
Above you can see the code for 2 buttons and one LED.

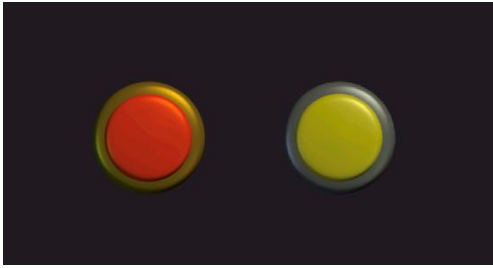
Initial case



Here is the initial situation of controlling the LED by 2 buttons where no one button was pressed, and the LED is switched off.

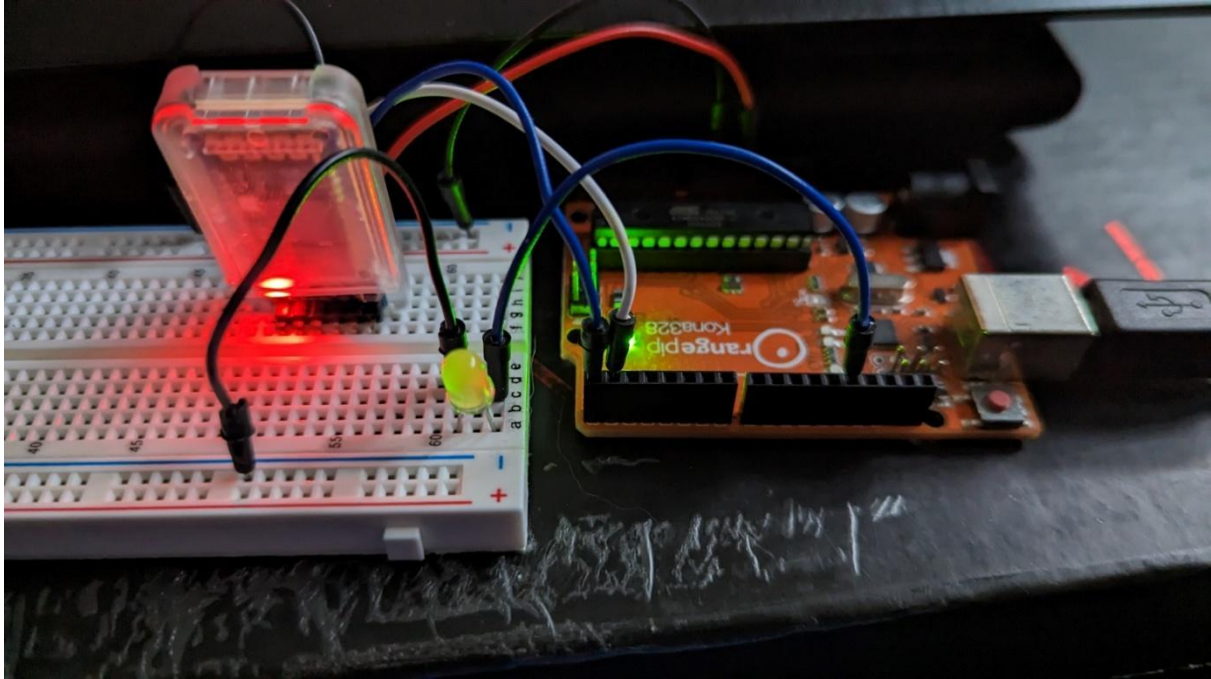
Green button pressed case.





Here you can see when we press the green button and release it the LED is switched on.

Yellow button pressed.



And in the last case when we press and release the yellow button the LED is turned off.



## This is the group 2 poster where we tried to include our knowledge about the case study

### Introduction

This case study provides an introduction to Robotics and Internet of Things (IoT) by emphasising the basics of electronic parts, sensors, and actuators. This module aims to make robotic systems more approachable and user-friendly for individuals of all skill levels and abilities. Additionally, the use of C programming using the Arduino IDE offers a versatile platform that is inclusive and accessible to a diverse range of learners.

### Aims & Objectives

This project aims to design two robot cars that can be operated via Bluetooth, with a focus on accessibility and inclusivity. One car will be controlled by an Android smartphone, while the other will be controlled by an iOS smartphone, ensuring that users of both platforms have equal access to the technology.

### Components Used/Background

- 2 Chassis basic kits:** Composed of acrylic chassis, wheels, motors, wires, breadboard. \*Unleaded soldered motors were used to promote safety, environmental responsibility and sustainable practices.
- 2 Arduino Nanos:** Small and versatile microcontroller boards compatible with the Arduino IDE and USB interface for programming and communication.
- 2 5V Batteries:** For the purpose of supplying current to the Arduino boards.
- 2 H-Bridges:** an electronic circuit that allows a voltage to be applied across a load.
- 1 HC-05 (Android) & 1 HC-16 (iOS) Bluetooth Modules:** Electronic devices that allow wireless communication between an Arduino board and other Bluetooth-enabled devices such as smartphones.

### Hardware Design & Construction

The cars were assembled, wheels were connected to the motors, the motor wires were soldered then connected to the H-Bridge driver.

The pins of the H-Bridge were connected to the Arduino as follows:  
B1A to Pin 4,  
B1B to Pin 5,  
GND to Pin GND  
VCC to Power 5V  
A1A to Pin 6  
A1B to Pin 7

The use of external power banks to supply the Arduino boards highlights a commitment to sustainability. By choosing rechargeable power sources, this project promotes environmentally-friendly practices and provides a cost-effective option for individuals from various socio-economic backgrounds.

For the Bluetooth modules, the pins connected to the Arduino were:  
RXD to Pin TXD  
TXD to Pin RXD  
GND to Pin GND  
VCC to Power 5V

*This picture features our Car number 1, a compact and agile Arduino Robot Car equipped with two wheels and controlled wirelessly by an Android smartphone.*

*This picture features our Car number 2, a compact and agile Arduino Robot Car equipped with four wheels and controlled wirelessly by an iOS smartphone.*

### Software Design & Development

In a brief explanation, our Arduino program receives Bluetooth data and controls the direction of the robot cars.

The program sets up digital pins 4, 5, 6, and 7 as outputs and defines five functions to control the movement of the cars: forward(), backward(), left(), right(), and stop\_all(). See some of the functions below:

```
void setup()
{
  digitalWrite(B1A, HIGH); //Send a signal to pin 4
  digitalWrite(B1B, HIGH); //Send a signal to pin 5
  digitalWrite(A1A, LOW); //Send a signal to pin 6
  digitalWrite(A1B, LOW); //Send a signal to pin 7
}

void forward()
{
  digitalWrite(B1A, LOW); //Send a signal to pin 4
  digitalWrite(B1B, HIGH); //Send a signal to pin 5
  digitalWrite(A1A, LOW); //Send a signal to pin 6
  digitalWrite(A1B, HIGH); //Send a signal to pin 7
}

void left()
{
  digitalWrite(B1A, LOW); //Send a signal to pin 4
  digitalWrite(B1B, LOW); //Send a signal to pin 5
  digitalWrite(A1A, HIGH); //Send a signal to pin 6
  digitalWrite(A1B, LOW); //Send a signal to pin 7
}

void right()
{
  digitalWrite(B1A, LOW); //Send a signal to pin 4
  digitalWrite(B1B, LOW); //Send a signal to pin 5
  digitalWrite(A1A, LOW); //Send a signal to pin 6
  digitalWrite(A1B, HIGH); //Send a signal to pin 7
}

void stop_all()
{
  digitalWrite(B1A, LOW); //Send a signal to pin 4
  digitalWrite(B1B, LOW); //Send a signal to pin 5
  digitalWrite(A1A, LOW); //Send a signal to pin 6
  digitalWrite(A1B, LOW); //Send a signal to pin 7
}
```

Note that, by providing clear and concise descriptions of the code, good comments promote accessibility and inclusivity by making it easier for individuals of different skill levels and backgrounds to understand the code, including those with visual or cognitive impairments. Additionally, well-documented code encourages collaboration and participation from underrepresented groups.

In the loop(), the program checks for incoming Bluetooth data and executes the corresponding function based on the received character (W, S, A, D). If no character is received, the function stop\_all() is called to stop the car. See picture below for reference.

```
void loop()
{
  if (Serial.available() > 0) { //Check if there are data on the serial port
    char c = Serial.read(); //Read the data and store it in the character variable c
    if (c == 'W') forward();
    if (c == 'S') backward();
    if (c == 'A') left();
    if (c == 'D') right();
    if (c == 'X') stop_all();
  }
}
```

To setup the Bluetooth connection it was quite simple. Once we attached the Bluetooth module to the Arduino, we ran a script to test if the Bluetooth was working, the same approach that we've done to check the Arduino Board at the beginning.

After tested, we then uploaded to the Arduino this final code which contains everything we needed to use the Bluetooth module as the way we control our cars.

### Phone HMI Via Bluetooth

Both cars have the same control configuration (W,D,S,A), the only difference their UI, which is designed specifically for the respective operating systems (iOS & Android) to ensure seamless UX for all users.

When the buttons on the interface are held down, they send an uppercase letter, which triggers the corresponding function. If none of the declared functions are matched, the "stop\_all" function is called, which stops the car. See below how our interfaces were arranged:

#### Bluetooth for Arduino (iOS UI)

*Both the arrows on the left and the Geometric Shapes on the right were set the same letters to give more options for whoever is controlling the car.*

#### Bluetooth Electronics (Android UI)

**Green button:** sends the uppercase **W**, which calls the function **forward()**.  
**Red button:** sends the uppercase **S**, which calls the function **backward()**.  
**Round button:** sends the uppercase **A**, which calls the function **left()**.  
**Square button:** sends the uppercase **D**, which calls the function **right()**.  
\*The **Lower case** equivalent is sent if **no button** is pressed, calling the function **stop\_all()**.

### Conclusion

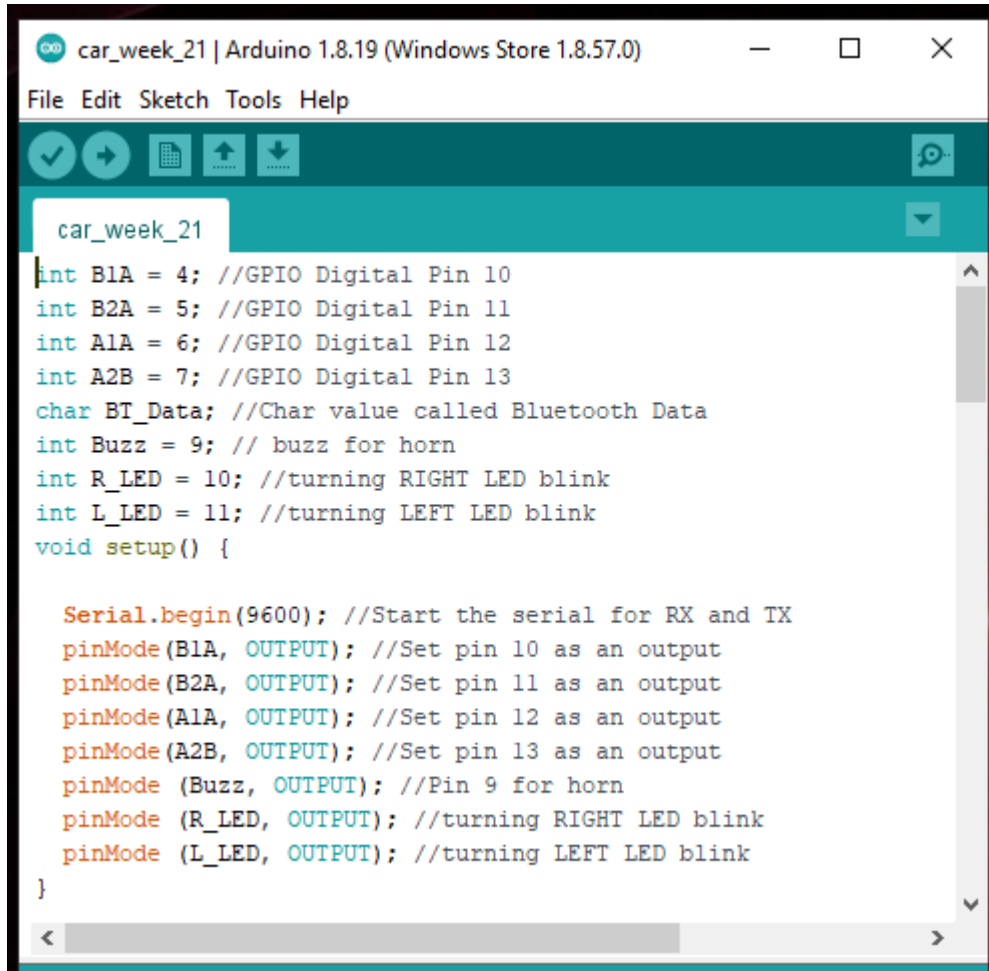
In conclusion, this case study was crucial for our understanding of Robotics and IoT fundamentals. It challenged us to approach problems from different angles, prioritising LSEP & inclusivity in every step of the process, from component assembly to programming and comprehending every function. Working in a smaller group taught us how to collaborate effectively and be as efficient as possible.

Despite facing numerous failures, stress, and anxiety, we ultimately achieved our goal for this project, which is what matters the most. In my opinion, this module was by far the most engaging and hands-on of the year, providing an exciting and interactive learning experience.

### Acknowledgements

We would like to thank all our tutors Dion, James and Richa for their support throughout the entire process and the insightful feedbacks and guidance that helped us complete this case study. Without this clear and patient guidance, we would not have been able to complete this project. Big thanks to group 6 as well for supporting us on the Robot Wars, we really appreciated it.

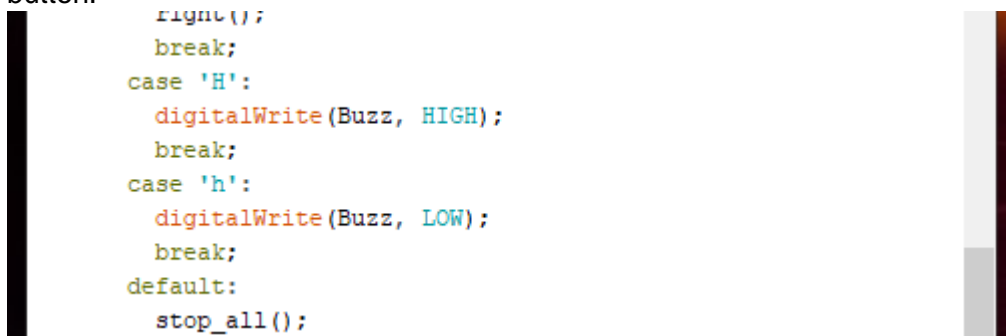
As one of additional features we chose to add a buzzer like a horn.

A screenshot of the Arduino IDE window titled "car\_week\_21 | Arduino 1.8.19 (Windows Store 1.8.57.0)". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu bar is a toolbar with icons for checking, running, uploading, and downloading. The sketch name "car\_week\_21" is displayed in a teal bar. The code editor shows the following code:

```
int B1A = 4; //GPIO Digital Pin 10
int B2A = 5; //GPIO Digital Pin 11
int A1A = 6; //GPIO Digital Pin 12
int A2B = 7; //GPIO Digital Pin 13
char BT_Data; //Char value called Bluetooth Data
int Buzz = 9; // buzz for horn
int R_LED = 10; //turning RIGHT LED blink
int L_LED = 11; //turning LEFT LED blink
void setup() {

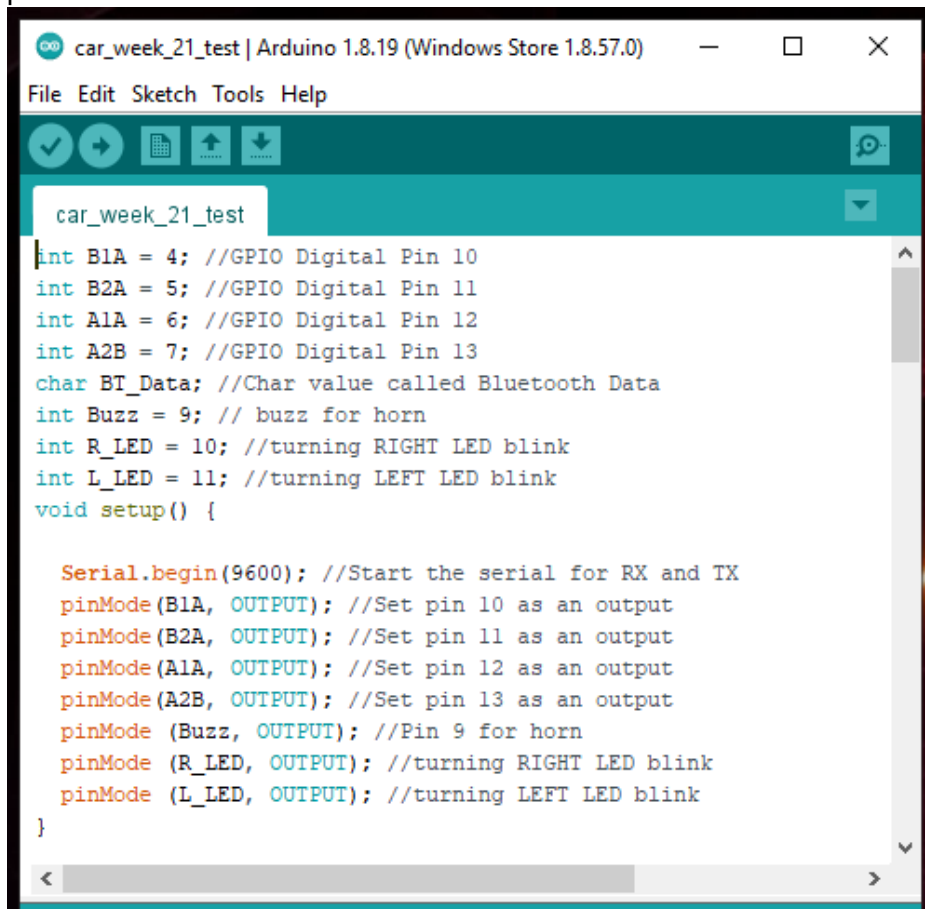
    Serial.begin(9600); //Start the serial for RX and TX
    pinMode(B1A, OUTPUT); //Set pin 10 as an output
    pinMode(B2A, OUTPUT); //Set pin 11 as an output
    pinMode(A1A, OUTPUT); //Set pin 12 as an output
    pinMode(A2B, OUTPUT); //Set pin 13 as an output
    pinMode (Buzz, OUTPUT); //Pin 9 for horn
    pinMode (R_LED, OUTPUT); //turning RIGHT LED blink
    pinMode (L_LED, OUTPUT); //turning LEFT LED blink
}
```

We connected the buzzer to pin 9 and wrote in the loop section two different cases for this button.

A screenshot showing a portion of the code in the Arduino IDE, specifically the loop() function. The code is as follows:

```
    right();
    break;
case 'H':
    digitalWrite(Buzz, HIGH);
    break;
case 'h':
    digitalWrite(Buzz, LOW);
    break;
default:
    stop_all();
```

The other additional feature we added two LED's as turning signal, which we connected to the pins 10 and 11.



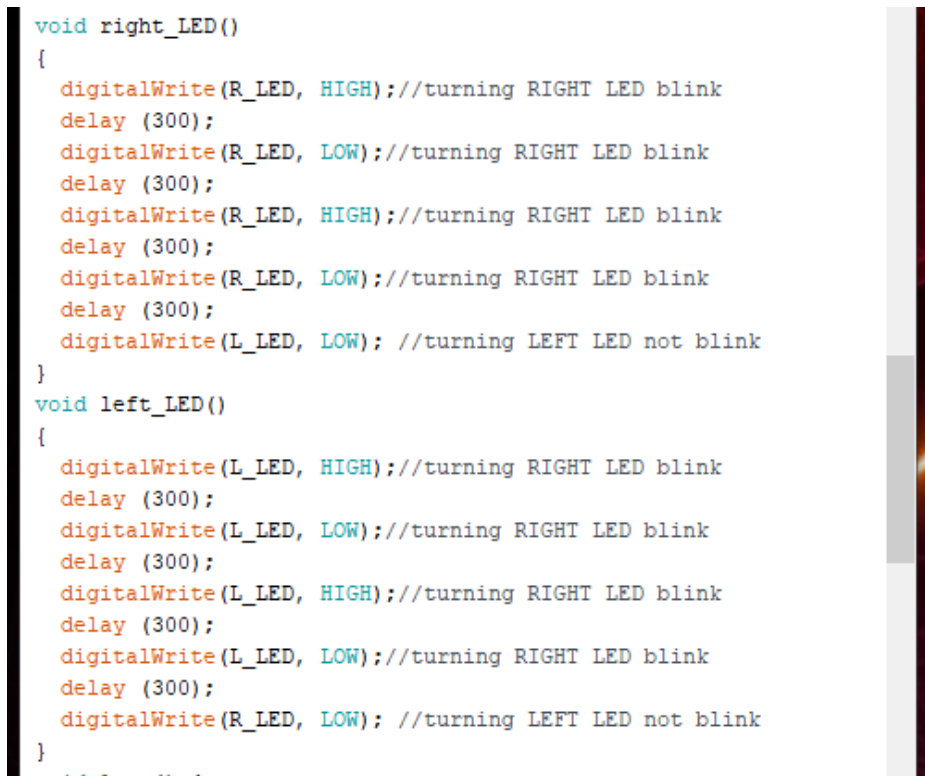
```
car_week_21_test | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help

car_week_21_test

int B1A = 4; //GPIO Digital Pin 10
int B2A = 5; //GPIO Digital Pin 11
int A1A = 6; //GPIO Digital Pin 12
int A2B = 7; //GPIO Digital Pin 13
char BT_Data; //Char value called Bluetooth Data
int Buzz = 9; // buzz for horn
int R_LED = 10; //turning RIGHT LED blink
int L_LED = 11; //turning LEFT LED blink
void setup() {

    Serial.begin(9600); //Start the serial for RX and TX
    pinMode(B1A, OUTPUT); //Set pin 10 as an output
    pinMode(B2A, OUTPUT); //Set pin 11 as an output
    pinMode(A1A, OUTPUT); //Set pin 12 as an output
    pinMode(A2B, OUTPUT); //Set pin 13 as an output
    pinMode(Buzz, OUTPUT); //Pin 9 for horn
    pinMode(R_LED, OUTPUT); //turning RIGHT LED blink
    pinMode(L_LED, OUTPUT); //turning LEFT LED blink
}
```

We add two more functions for each LED and we added this function to the case buttons.



```
void right_LED()
{
    digitalWrite(R_LED, HIGH); //turning RIGHT LED blink
    delay(300);
    digitalWrite(R_LED, LOW); //turning RIGHT LED blink
    delay(300);
    digitalWrite(R_LED, HIGH); //turning RIGHT LED blink
    delay(300);
    digitalWrite(R_LED, LOW); //turning RIGHT LED blink
    delay(300);
    digitalWrite(L_LED, LOW); //turning LEFT LED not blink
}

void left_LED()
{
    digitalWrite(L_LED, HIGH); //turning RIGHT LED blink
    delay(300);
    digitalWrite(L_LED, LOW); //turning RIGHT LED blink
    delay(300);
    digitalWrite(L_LED, HIGH); //turning RIGHT LED blink
    delay(300);
    digitalWrite(L_LED, LOW); //turning RIGHT LED blink
    delay(300);
    digitalWrite(R_LED, LOW); //turning LEFT LED not blink
}
```

```
    break;
case 'A':
    left();
    left_LED();
    break;
case 'D':
    right();
    right_LED();
    break;
```