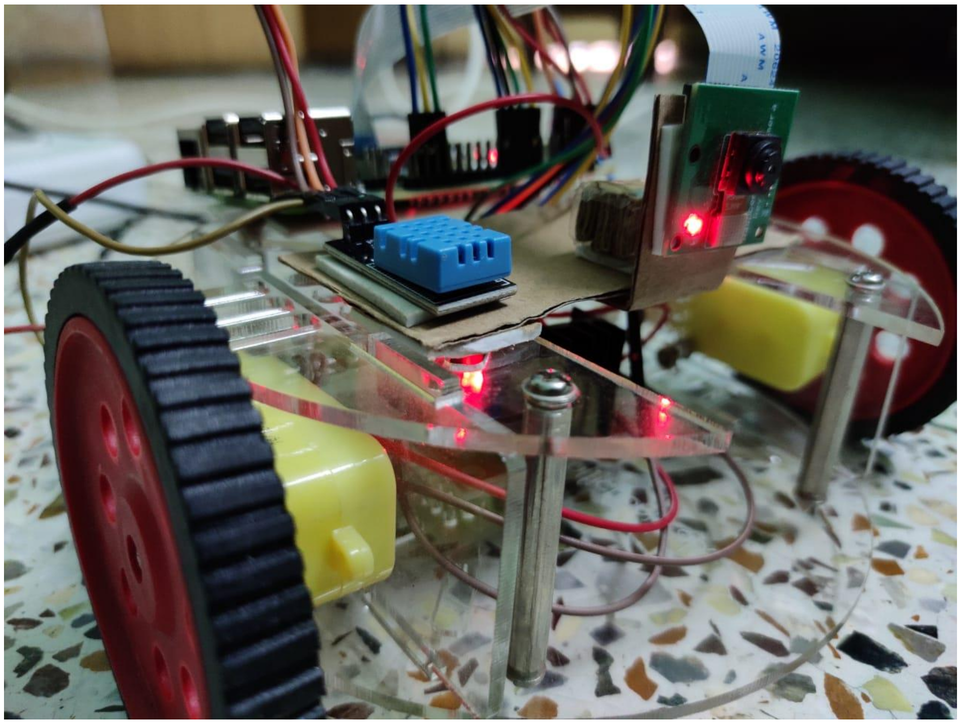
TARP – CSE3999

REMOTE SURVEILLANCE BOT

Mayukh Ghosh (18BCE0417)
Saksham Goyal (18BCE2196)
Rohan Allen (18BCI0247)



The video demo of the bot can be seen in the link below:

https://drive.google.com/file/d/1f93gnEgdF9KD89zgha-H9lmyZD8qhhPZ/view?usp=sharing

(The code screen-shots bot’s image are to the end of the document)

HARDWARE REQUIRED:

Raspberry Pi
The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

Raspberry Pi camera board
The Camera Module can be used to take high-definition video, as well as stills photographs. ... It supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

L298 Motor Driver Module
It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors.

DHT11 Sensor
The DHT11 is a basic, ultra-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).

Bot chasse
Chassis is very essential in robots as well as many mechanical devices. Robot chassis is particularly designed for robots and other mechanical devices. These accessories handle PCB, components and parts that are interfaced and connected to it.

2 DC motors
Simplest type of motors which work on direct current.

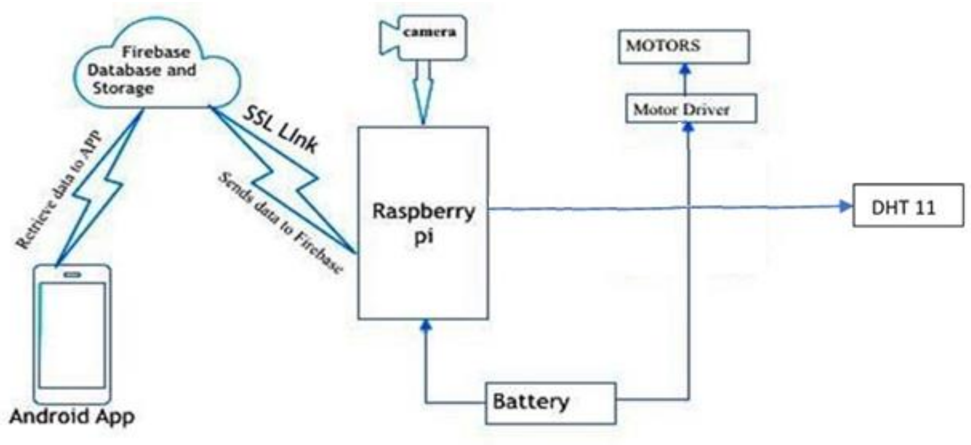
SOFTWARE REQUIRED:

Firebase Realtime database
Google's Firebase is a safe and secure cloud platform to store and host data. On Firebase, secure user authentication is performed so that only the authorized person has access to the data.

Android Studio
Built on JetBrains' IntelliJ IDEA software and customized exclusively for Android development, Android Studio is the official integrated development environment for Google's Android operating system.

TensorFlow
TensorFlow is a machine learning and artificial intelligence software library that is free and open-source. It can be used for a variety of applications, but it focuses on deep neural network training and inference.

OVERVIEW OF THE PRODUCT:



We will develop an android application using android studio that will be like the remote of the bot. We will give it joystick-like keys through which the bot can be controlled. The rotation/ movement of the keys will change the data (angle and magnitude) in the cloud storage, and through cloud functions, any change in the data will trigger a data transfer to the raspberry pi.
The raspberry pi will process the data and accordingly send signals to arduino to move the bot via the motor drivers.

The bot is fixed with some sensors like DHT11 and camera. The camera captures the surroundings and sends the data to raspberry pi. In the raspberry the date is processed using the algorithm mentioned below and is uploaded to the cloud.
The temperature and humidity details from the DHT11 sensor is received by arduino and is sent to raspberry pi which is uploaded to the cloud from there.
The android app receives the temperature details and humidity details of the terrain from the cloud and it is displayed to the user. The image is uploaded to the firebase’s Realtime database in the form of a string of base64. The application converts it back to the image from that format. Then the ML analysis is done over that image to give users more details about the terrain. And the image is shown to the user with the ML analysis.

ALGORITHM TO LIVE STREAM VIDEO VIA FIREBASE:
Firebase is a cloud platform by google. It is secure and scalable. It has many components, but we will be using its real time database. The real time database is designed to stream real time text data in the cloud platform. We in our project will use it in an innovative way to stream live video.
1. The camera captures the video of the surroundings and gives it to raspberry pi.
2. Raspberry pi, using base64 library takes frames of those videos and converts them to text format
3. The text format used is the base 64 string format. The string is then uploaded to the Realtime database.
4. The android app fetches the base 64 string from the Realtime database and converts it into image format.
5. The image is then analysed using the ML algorithm as described below.
6. The image along with the analysis is then displayed into an image viewer in the android app.
7. By frequently repeating this process (1-6) we will give the illusion of live video display to the user operating the bot.

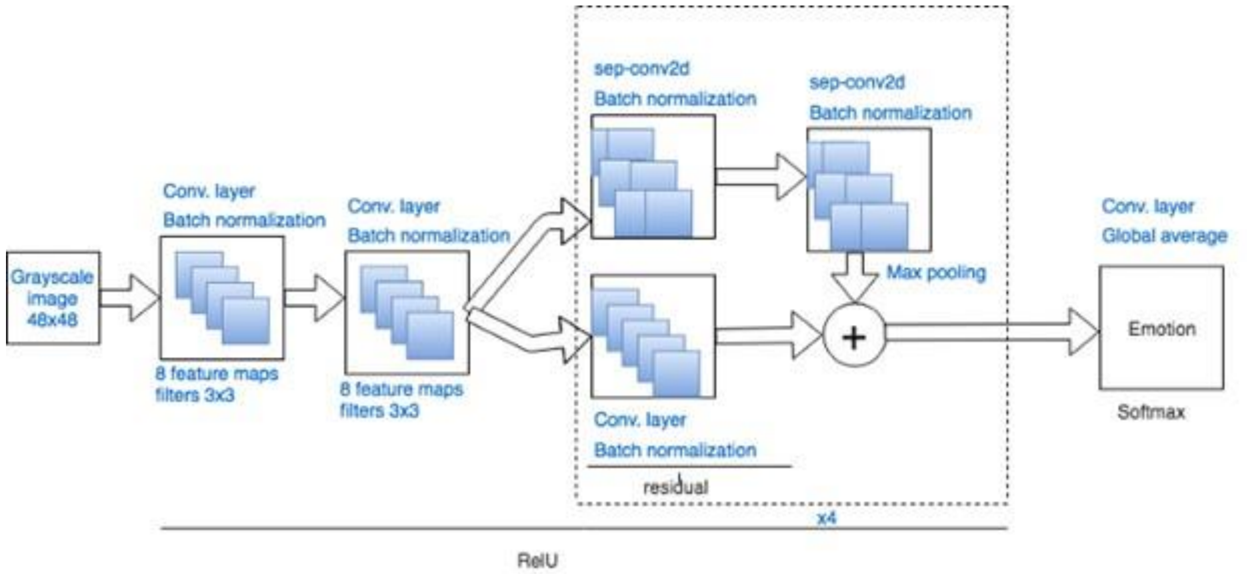
ALGORITHM TO CONTROL THE BOT REMOTELY OVER CLOUD:
For this functionality also we will be using the firebase. We will take data from the app and send it to the bot for its proper

functioning.
1. Input the position of the joystick from the android app.
2. Upload the positional value of the joystick if the position changes into the Realtime database of Firebase.
3. The data is triggered via a cloud function whenever it gets changed to the raspberry pi. Raspberry Pi receives the data from the cloud and accordingly sends instructions to the arduino to operate the dc motors via the motor driver.
4. In exactly the same way the DHT 11 sensor sends back temperature and humidity data to the android app via the cloud platform.

ALGORITHM TO DETECT HOSTILE ENVIRONMENT USING ML:

Deep learning for Emotion Recognition

With respect to engineering of the CNN, a concise hypothetical presentation is given. In the wake of attempting a few models, the one proposed in ended up being the most incredible as far as precision and preparing effectiveness. In Figure 2 the engineering is addressed.



A few developments are done in this Convolutional Neural Network with the goal that a precision improvement is accomplished.
Above all else, is the idea of remaining modules. Remaining modules are a method to facilitate the preparation of organizations, and they change the ideal planning between two ensuing layers,

so, the learned provisions become the distinction of the first component map and the ideal features.
Also, profundity savvy distinct convolution which is a type of factorized convolutions which factorize a standard one into a profundity astute convolution and a 1 x 1 convolution called a point-wise convolution. Accordingly, there are two fundamentals layers in every convolution, which intend to isolate the spatial cross-connections from the channel cross-relationships.
Profundity insightful convolutions are utilized to apply a solitary channel for each information channel (input profundity).
Point-wise convolution, a basic 1 x 1 convolution, is then used to make a straight blend of the yield of the profundity savvy layer. Despite they are incredibly productive contrast with standard convolutions, it doesn't consolidate the sifted channels to make new provisions. That is the place where the new layer comes up, the point-wise layer is utilized to register a direct blend of the yields of the profundity savvy convolution. More or less, profundity savvy divisible convolutions decrease the calculation regarding the standard convolutions, with the goal that the preparation stage is quicker.
Thirdly, as the preparation interaction with the past network was famously hard and subsequently soaked with a nonlinearity, otherwise called inner covariate shift, an answer found is normalizing the contributions of the layers. This is called Batch-standardization, and it additionally goes about as a regularizer staying away from in specific cases the utilization of dropouts. This method plays out a simple activity which is applied to enactment x over a scaled down clump.
At long last the idea of regularization applied in certain layers. Regularizers focus on sum up the conduct of the net. Explicitly in this engineering, along with other currently referenced, l2-regularization (otherwise known as weight rot) will be executed in certain layers. The possibility of L2 regularization is to add an additional term to the expense work, called the regularization term. For instance, utilizing the cross-entropy work the speculation term is added as follows:

$$C = -\frac{1}{n}\sum_{x}[y \ln y' + (1-y)\ln(1-y')] + \frac{\lambda}{2n}\sum_{w} w^2$$

The initial term relates to the actual capacity, and the subsequent one is the term, wherein the amount of the squares of the multitude of loads in the organization is done, scaled by a factor, where λ is the regularization boundary. For this situation the scale factor utilized is 0.01.
Hence, our last design, as displayed in Figure 2 is a completely convolutional neural organization comprising of 2 convolutional layers of 8 components. ReLu actuation, 3x3 with a portion regularizer, trailed by a group standardization. Then, at that point there are four modules wherein a remaining is carried out (Convolutional layer, 1x1, trailed by a bunch standardization) and added to the distinct convolutional layer, 3x3 with portion regularization, trailed by a group standardization also. The size of these convolutional pieces are expanding in every module, beginning in 16 and finishing in 128. At last, the actual arrangement is brought out through a convolutional layer with the size equivalent to the quantity of feelings to be anticipated, trailed by a Global normal pooling layer with Softmax enactment.

ABOUT ANDROID APP:

The app will be developed in android studio. It will have a very user-friendly interface and simplistic design. It will have two joystick buttons to control the bot.
And a screen in the middle to show the live video from the environment. The panel will also give the details of the temperature and humidity. If some human faces are detected in the video, it will also alert the user with its UI, as it will prove to be an important feature in rescue, spying and explorations operations.

INNOVATION / NOVELTY:

Unlike traditional surveillance bots our bot uses WIFI Technology for communication purposes. This not only offers a much superior range as compared to Bluetooth or radio frequency enabled devices but also is more secure. Movement is not hindered or restricted in any fashion due to the extremely versatile and sleek Arduino UNO motor. It also incorporates cutting edge Machine Learning algorithms to detect human facial features and also possess the state-of-the-art DHT11 sensor which relays real time temperature and humidity data, something which is extremely invaluable. The key challenge in prior surveillance bots made with Raspberry Pi or Arduino was establishing a secure connection between the bot and the controlling system. This difficulty is solved by establishing a link between the bot and the controlling device in order to transfer control signals and video streaming using Firebase Real-Time Database. Google's Firebase is a safe and secure cloud platform. To operate the bot, we will use an app created in Android Studio. In Android Studio, we're building an app with the flutter SDK framework. We will develop the app to communicate with Firebase and deliver control signal information to the robot. In addition, the video data must be retrieved and stored in an app. On Firebase, secure user authentication is performed so that only the authorized person has access to the intelligence information and can control our bot.

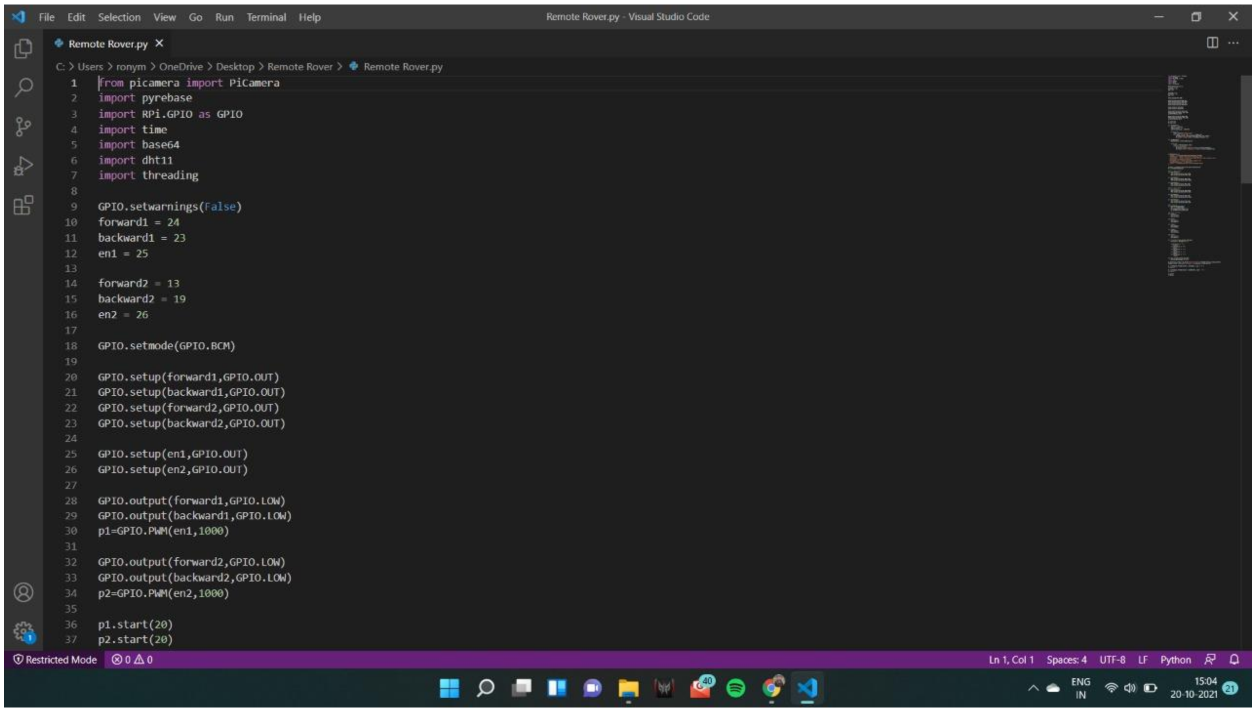
Project Outcome

The final objective of our project is to create a fully functional product, i.e., a movable can move thus can cover all the areas in contrast to CCTVs remote surveillance bot with a tiny Camera mounted on it. This can be used for numerous tasks, varying from spying, reconnaissance, information gathering, terrain exploration, border patrolling, etc. To eliminate the range problem of modern-day bots we will be controlling the bot over the internet. The bot can thus cover all the areas in contrast to CCTVs. This bot will be operated on the Wi-Fi network, thereby giving it, a much greater mobility of range and it can be controlled from anywhere as long as they are communicating on the same network, which is a great convenience. This bot does not require radio signals or Bluetooth signals, hence there is no range limit as is the case for the majority of the products in the market. The bot has a camera that will give us the live stream of the location. This bot can be useful for home security. This bot can even be deployed in army and can be operated in any territory where internet is available. This will be a low-cost bot as all the components used, are extremely cheap and easily available, thereby adding considerable value to the product. Raspberry-pi is at the heart of the system, and it is responsible for all of the system's operations. We will also create an Android based application which will be used to control the bot. It will have buttons to control the camera and the range of movement of the bot. It can move in any of the four cardinal directions namely, north, south, east and west, thereby enabling three sixty-degree range of mobility. The application will also be embedded with machine learning algorithms

which will be beneficial in recognizing human faces very easily and detect the hostility level of different approaching targets.
The Raspberry Pi is being utilized as the core controller for the surveillance robot, and a USB camera is being used to capture video for live streaming. The Raspberry Pi was connected to the camera. Arduino uno was used to control the forward, backward, right, and left motions. The L293D motor driver was used to operate the motor from the Arduino. The Raspberry Pi sends control signals to the Arduino. The information is sent to Arduino via GPIO pins on the Raspberry Pi. The 1800mAh LiPO battery that powers the robot has a 5v output. The motor was powered by a 12v external battery. The key challenge in prior surveillance bots made with Raspberry Pi or Arduino was establishing a secure connection between the bot and the controlling system. This difficulty is solved by establishing a link between the bot and the controlling device in order to transfer control signals and video streaming using Firebase Real-Time Database. Google's Firebase is a safe and secure cloud platform. Firebase offers a real-time database that may be used to broadcast data in real-time. Data that has been posted to a cloud server can be retrieved in real-time using this method. Firebase provides a variety of tools to programmers, including RTD, Information Storage, and Machine Learning Kit. Real-time databases, or RTDs, are a type of database that is updated in real-time. Firebase is one of the greatest platforms for creating a database that can fetch information in real - time without causing any delays in the end system. Firebase is secure since it establishes an SSL connection. As a result, the connection between the host and the server is secured. We're using a Raspberry Pi to get live footage from a USB camera in this example. The data is then pre-processed and encoded before being sent to a real-time Firebase database and retrieved by an Android app using our algorithmic method. We're using an android phone as a remote, and we're using an android app that we built with Android Studio and the Flutter SDK. The control signal was received from Firebase using a Python program on the Raspberry Pi, and the signal data was sent to Arduino through GPIO. The Arduino then processes the control signal and instructs the robot's motors to move. Raspberry Pi is a single-board computer chip that runs on a 1.2GHz Broadcom processor and contains 4GB of RAM. It also features 40 GPIO pins for connecting to sensors and other devices. As the robot's core controller, we're employing a Raspberry Pi. A USB camera and an Arduino were used to connect the Raspberry Pi. The core application was written in Python 2.7 and runs on the Raspbian Jessie operating system. The Raspberry Pi must be connected to the internet in order to interact with the Firebase cloud. To connect to the internet, we used the Raspberry Pi's built-

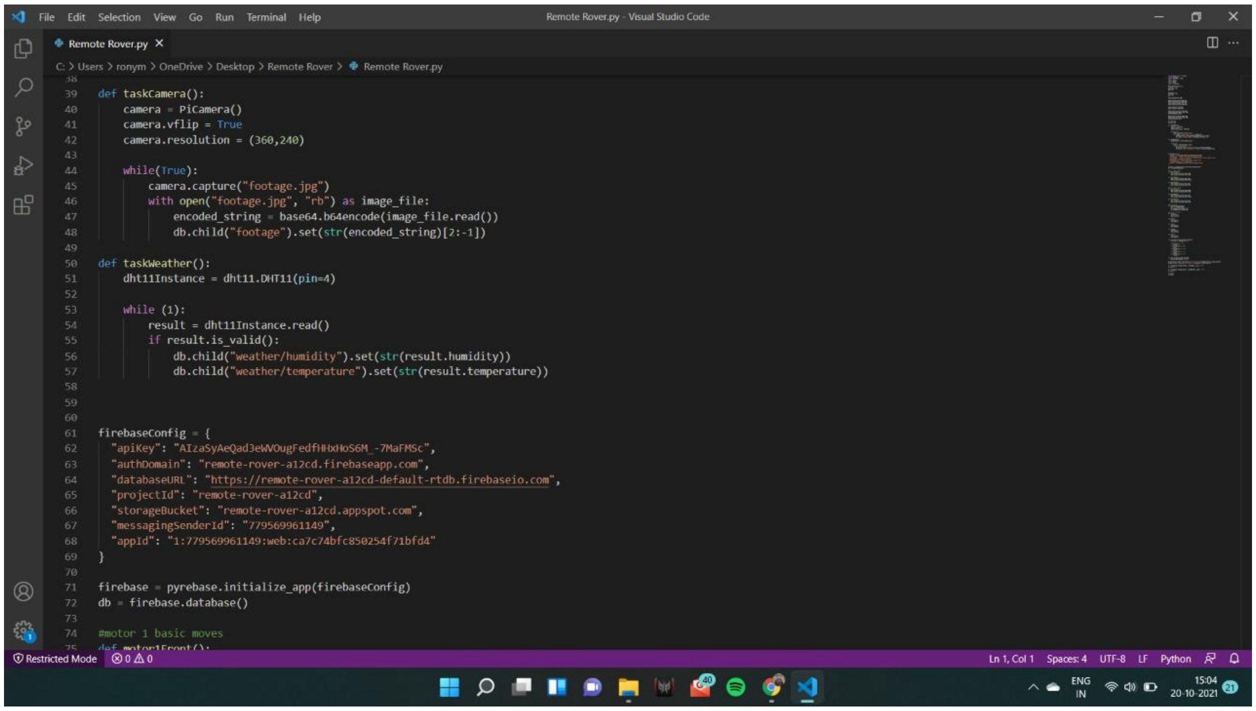
in WIFI card. We need more computing power than an Arduino or other microcontroller can provide, so we opt with the Raspberry Pi. Arduino is an Atmega microcontroller-based development board. Even while the Raspberry Pi has a fast processor, it isn't fast enough to do numerous tasks at once. As a result, we relied solely on the Arduino to control the mobility and the bmp180 sensor. The Arduino is in charge of the robot's movements. GPIO pins of the Raspberry Pi are used to send input to the Arduino. The Arduino then processes the information and sends it to the L293D motor driver, which turns on the appropriate motors. To operate the bot, we will use an app created in Android Studio. In Android Studio, we're building an app with the flutter SDK framework. We will use the Dart programming language to develop the app to communicate with Firebase and deliver control signal information to the robot. In addition, the video data must be retrieved and stored in an app. On Firebase, secure user authentication is performed so that only the authorized person has access to the intelligence information and can control our bot.

Code:


File Edit Selection View Go Run Terminal Help
Remote Rover.py - Visual Studio Code
Remote Rover.py
C: > Users > ronym > OneDrive > Desktop > Remote Rover > Remote Rover.py
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 LF Python

```python
from picamera import PiCamera
import pyrebase
import RPi.GPIO as GPIO
import time
import base64
import dht11
import threading

GPIO.setwarnings(False)
forward1 = 24
backward1 = 23
en1 = 25

forward2 = 13
backward2 = 19
en2 = 26

GPIO.setmode(GPIO.BCM)

GPIO.setup(forward1,GPIO.OUT)
GPIO.setup(backward1,GPIO.OUT)
GPIO.setup(forward2,GPIO.OUT)
GPIO.setup(backward2,GPIO.OUT)

GPIO.setup(en1,GPIO.OUT)
GPIO.setup(en2,GPIO.OUT)

GPIO.output(forward1,GPIO.LOW)
GPIO.output(backward1,GPIO.LOW)
p1=GPIO.PWM(en1,1000)

GPIO.output(forward2,GPIO.LOW)
GPIO.output(backward2,GPIO.LOW)
p2=GPIO.PWM(en2,1000)

p1.start(20)
p2.start(20)
```

Code to include the heard files and configure the GPIO pins

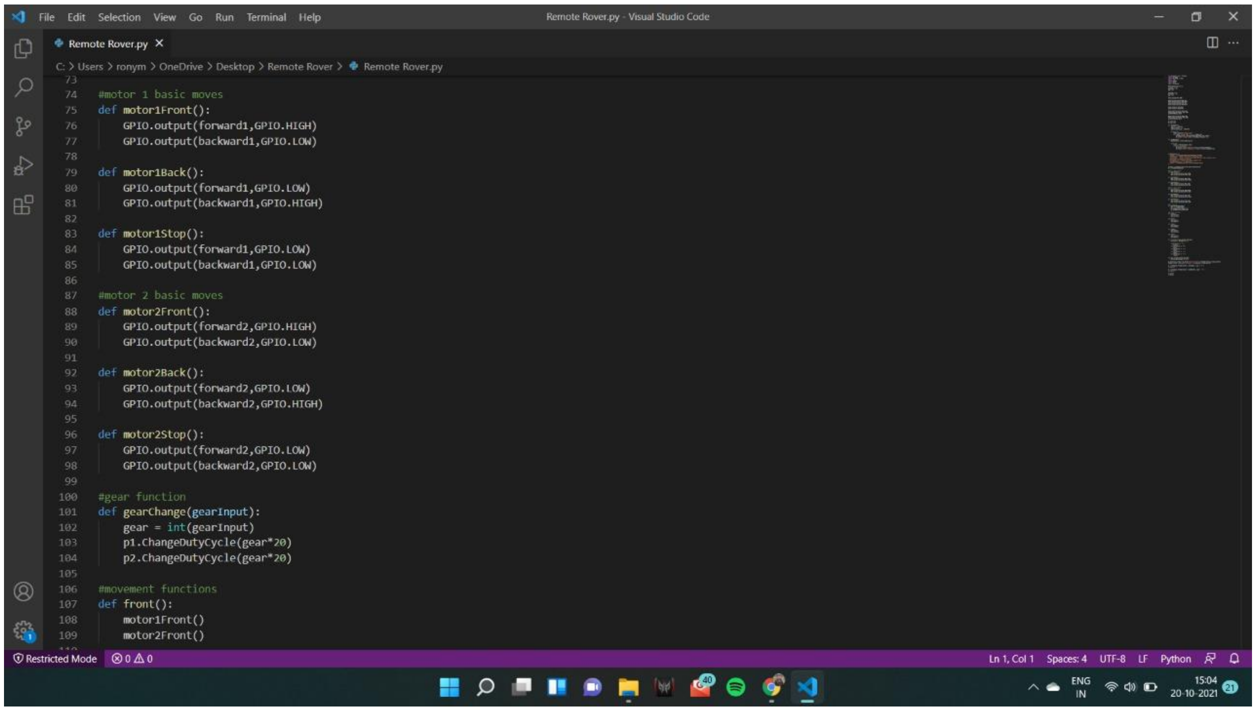


```python
def taskCamera():
    camera = PiCamera()
    camera.vflip = True
    camera.resolution = (360,240)

    while(True):
        camera.capture("footage.jpg")
        with open("footage.jpg", "rb") as image_file:
            encoded_string = base64.b64encode(image_file.read())
            db.child("footage").set(str(encoded_string)[2:-1])

def taskWeather():
    dht11Instance = dht11.DHT11(pin=4)

    while (1):
        result = dht11Instance.read()
        if result.is_valid():
            db.child("weather/humidity").set(str(result.humidity))
            db.child("weather/temperature").set(str(result.temperature))



firebaseConfig = {
  "apiKey": "AIzaSyAeQad3eWVOugFedfHHxHoS6M_-7MaFMSc",
  "authDomain": "remote-rover-a12cd.firebaseapp.com",
  "databaseURL": "https://remote-rover-a12cd-default-rtdb.firebaseio.com",
  "projectId": "remote-rover-a12cd",
  "storageBucket": "remote-rover-a12cd.appspot.com",
  "messagingSenderId": "779569961149",
  "appId": "1:779569961149:web:ca7c74bfc850254f71bfd4"
}

firebase = pyrebase.initialize_app(firebaseConfig)
db = firebase.database()

#motor 1 basic moves
```

In the above code snippet, the first function does the functionality to capture the image, convert it into the base64 string and upload it into firebase

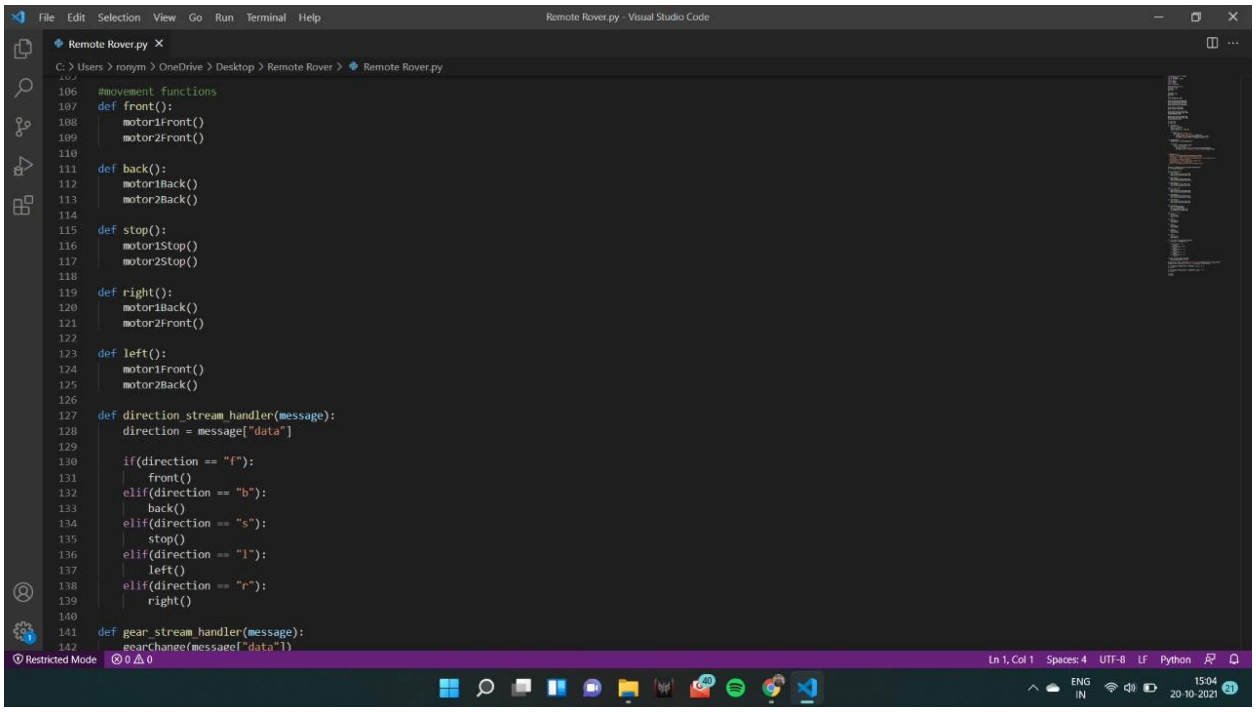
The second function gets the info of humidity and temperature from the sensor and uploads it to firebase

‘firebaseConfig’ variable stores the details to connect the project with the firebase real-time database.


Remote Rover.py - Visual Studio Code
C: > Users > ronym > OneDrive > Desktop > Remote Rover > Remote Rover.py
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 LF Python

```python
#motor 1 basic moves
def motor1Front():
    GPIO.output(forward1,GPIO.HIGH)
    GPIO.output(backward1,GPIO.LOW)

def motor1Back():
    GPIO.output(forward1,GPIO.LOW)
    GPIO.output(backward1,GPIO.HIGH)

def motor1Stop():
    GPIO.output(forward1,GPIO.LOW)
    GPIO.output(backward1,GPIO.LOW)

#motor 2 basic moves
def motor2Front():
    GPIO.output(forward2,GPIO.HIGH)
    GPIO.output(backward2,GPIO.LOW)

def motor2Back():
    GPIO.output(forward2,GPIO.LOW)
    GPIO.output(backward2,GPIO.HIGH)

def motor2Stop():
    GPIO.output(forward2,GPIO.LOW)
    GPIO.output(backward2,GPIO.LOW)

#gear function
def gearChange(gearInput):
    gear = int(gearInput)
    p1.ChangeDutyCycle(gear*20)
    p2.ChangeDutyCycle(gear*20)

#movement functions
def front():
    motor1Front()
    motor2Front()
```

The above code snippet defines the basic motor movements of the two DC motors i.e forward, backward and stop

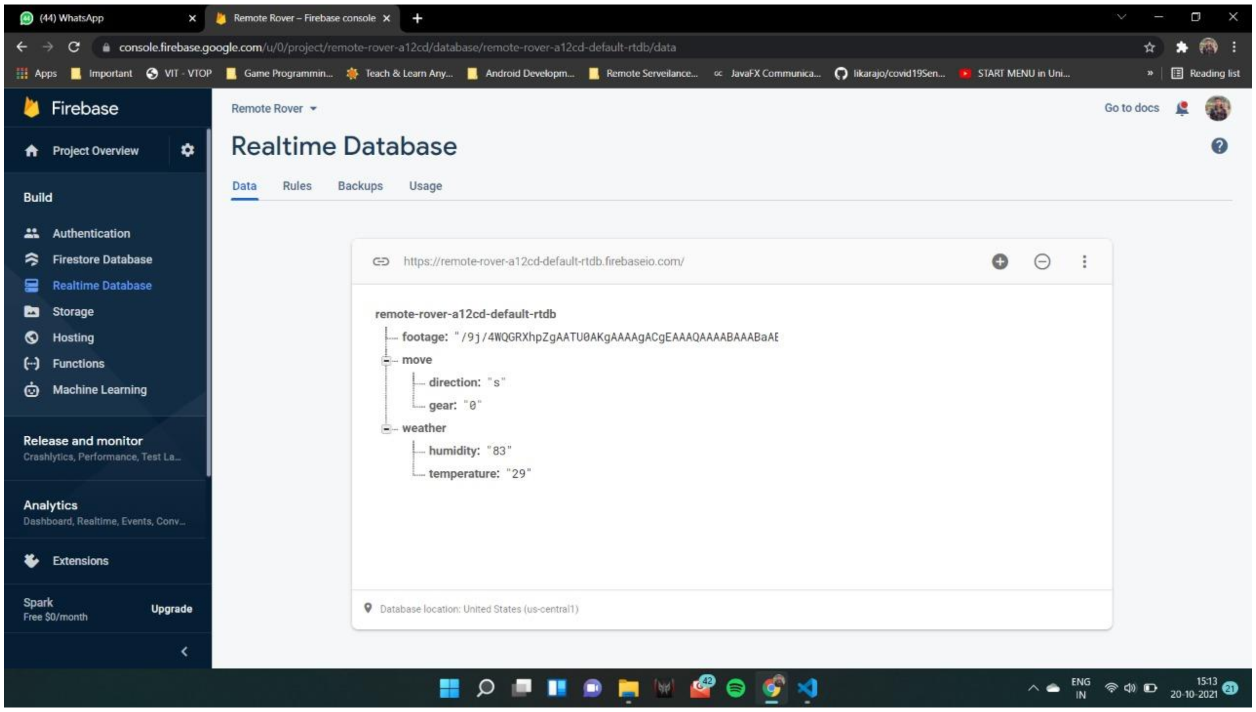

Remote Rover.py - Visual Studio Code
C: > Users > ronym > OneDrive > Desktop > Remote Rover > Remote Rover.py
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 LF Python

```python
#movement functions
def front():
    motor1Front()
    motor2Front()

def back():
    motor1Back()
    motor2Back()

def stop():
    motor1Stop()
    motor2Stop()

def right():
    motor1Back()
    motor2Front()

def left():
    motor1Front()
    motor2Back()

def direction_stream_handler(message):
    direction = message["data"]

    if(direction == "f"):
        front()
    elif(direction == "b"):
        back()
    elif(direction == "s"):
        stop()
    elif(direction == "l"):
        left()
    elif(direction == "r"):
        right()

def gear_stream_handler(message):
    gearChange(message["data"])
```

The above code snippet uses the different combination of the basic movements of the motors to result into movements of the bot.

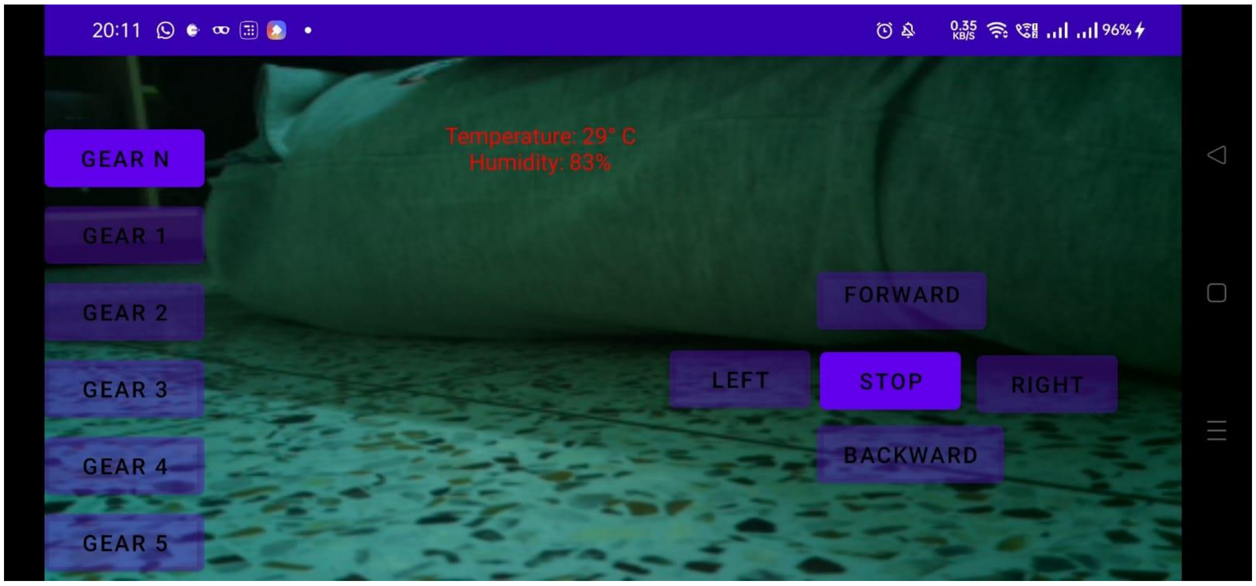
The ‘direction stream handler’ and ‘gear stream handler’ handles any change to the gear and direction branch of the firebase.

The code has been written in such a way that all the 4 cores of Raspberry Pi 3 is used simultaneously to get the maximum efficiency and least lag.

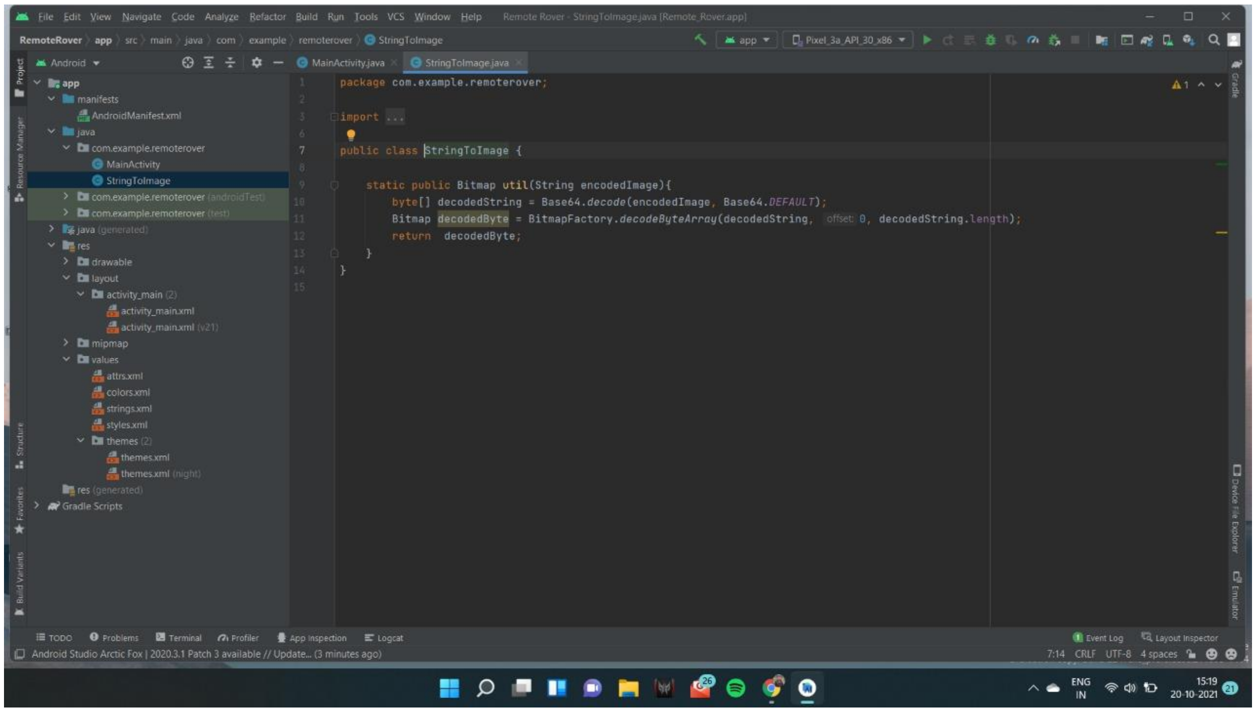


```
https://remote-rover-a12cd-default-rtdb.firebaseio.com/

remote-rover-a12cd-default-rtdb
  footage: "/9j/4WQGRXhpZgAATU0AKgAAAAgACgEAAAQAAAABAAABaAE
  move
    direction: "s"
    gear: "0"
  weather
    humidity: "83"
    temperature: "29"

Database location: United States (us-central1)
```

The above screenshot is of the real-time database of the firebase.

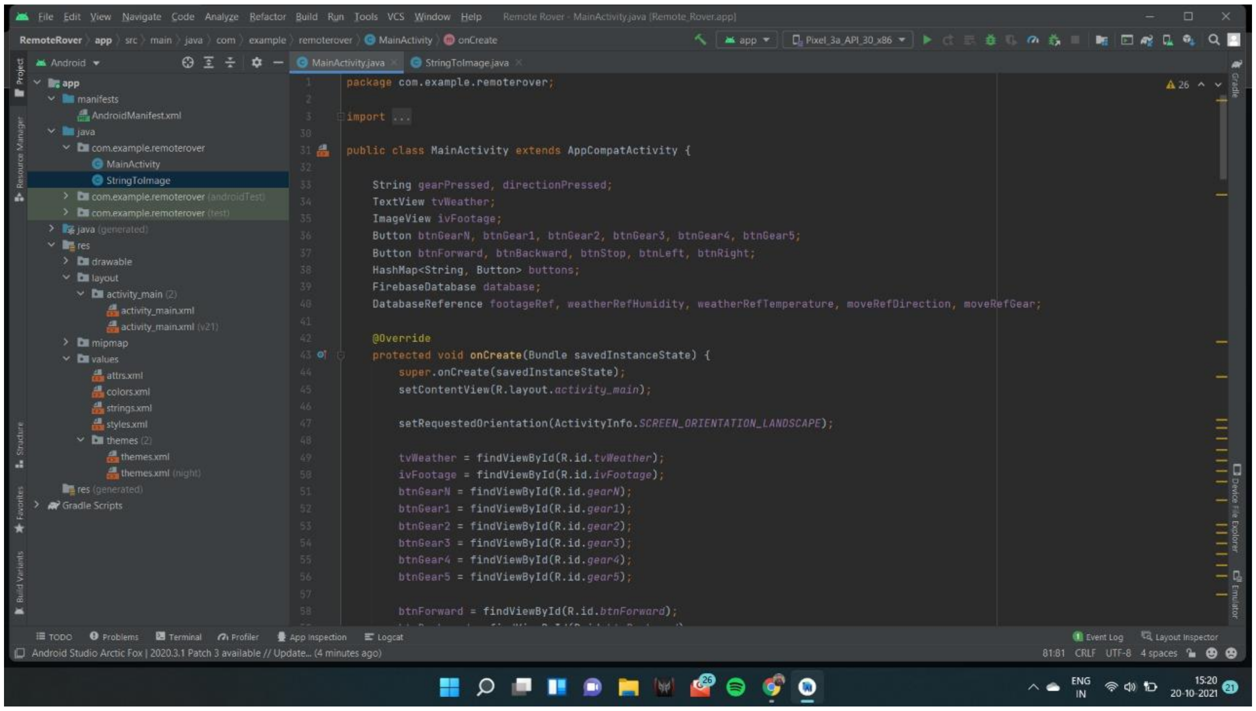


The screenshot of the android app



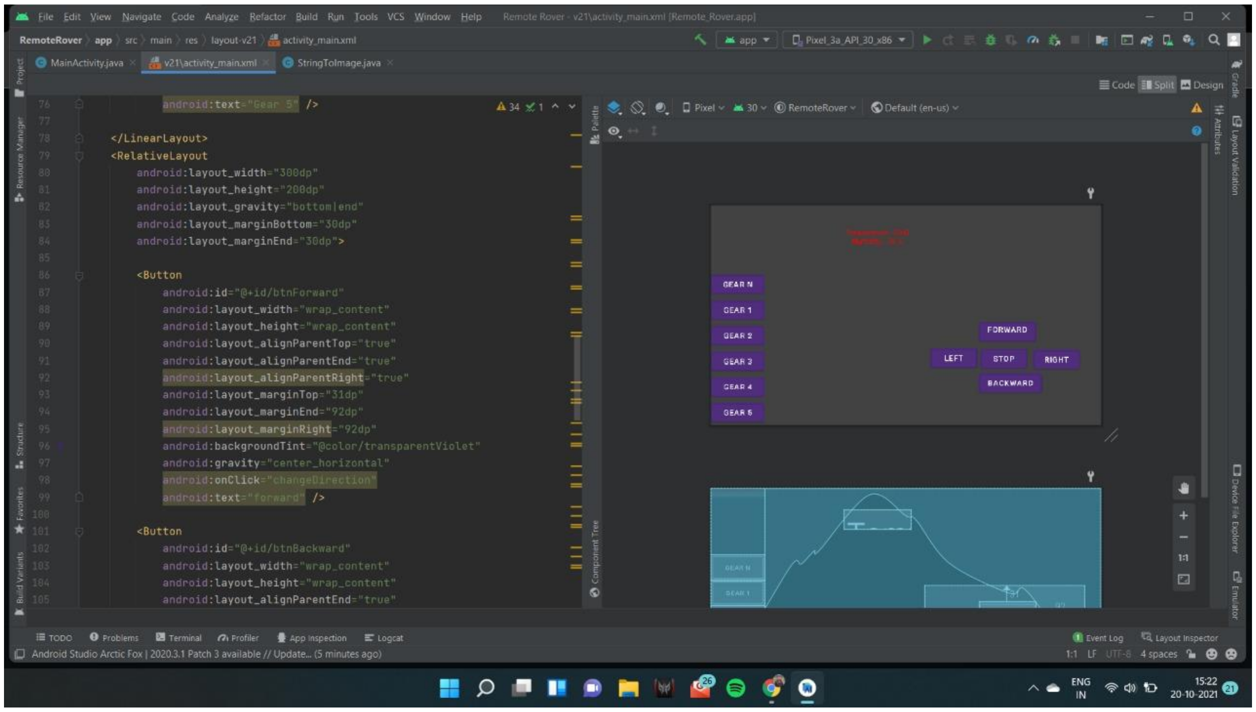
```
package com.example.remoterover;

import ...

public class StringToImage {

    static public Bitmap util(String encodedImage){
        byte[] decodedString = Base64.decode(encodedImage, Base64.DEFAULT);
        Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString, 0, decodedString.length);
        return  decodedByte;
    }
}
```

The above code snippet the string it fetches from firebase to image for the android app



```
package com.example.remoterover;

import ...

public class MainActivity extends AppCompatActivity {

    String gearPressed, directionPressed;
    TextView tvWeather;
    ImageView ivFootage;
    Button btnGearN, btnGear1, btnGear2, btnGear3, btnGear4, btnGear5;
    Button btnForward, btnBackward, btnStop, btnLeft, btnRight;
    HashMap<String, Button> buttons;
    FirebaseDatabase database;
    DatabaseReference footageRef, weatherRefHumidity, weatherRefTemperature, moveRefDirection, moveRefGear;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

        tvWeather = findViewById(R.id.tvWeather);
        ivFootage = findViewById(R.id.ivFootage);
        btnGearN = findViewById(R.id.gearN);
        btnGear1 = findViewById(R.id.gear1);
        btnGear2 = findViewById(R.id.gear2);
        btnGear3 = findViewById(R.id.gear3);
        btnGear4 = findViewById(R.id.gear4);
        btnGear5 = findViewById(R.id.gear5);

        btnForward = findViewById(R.id.btnForward);
```

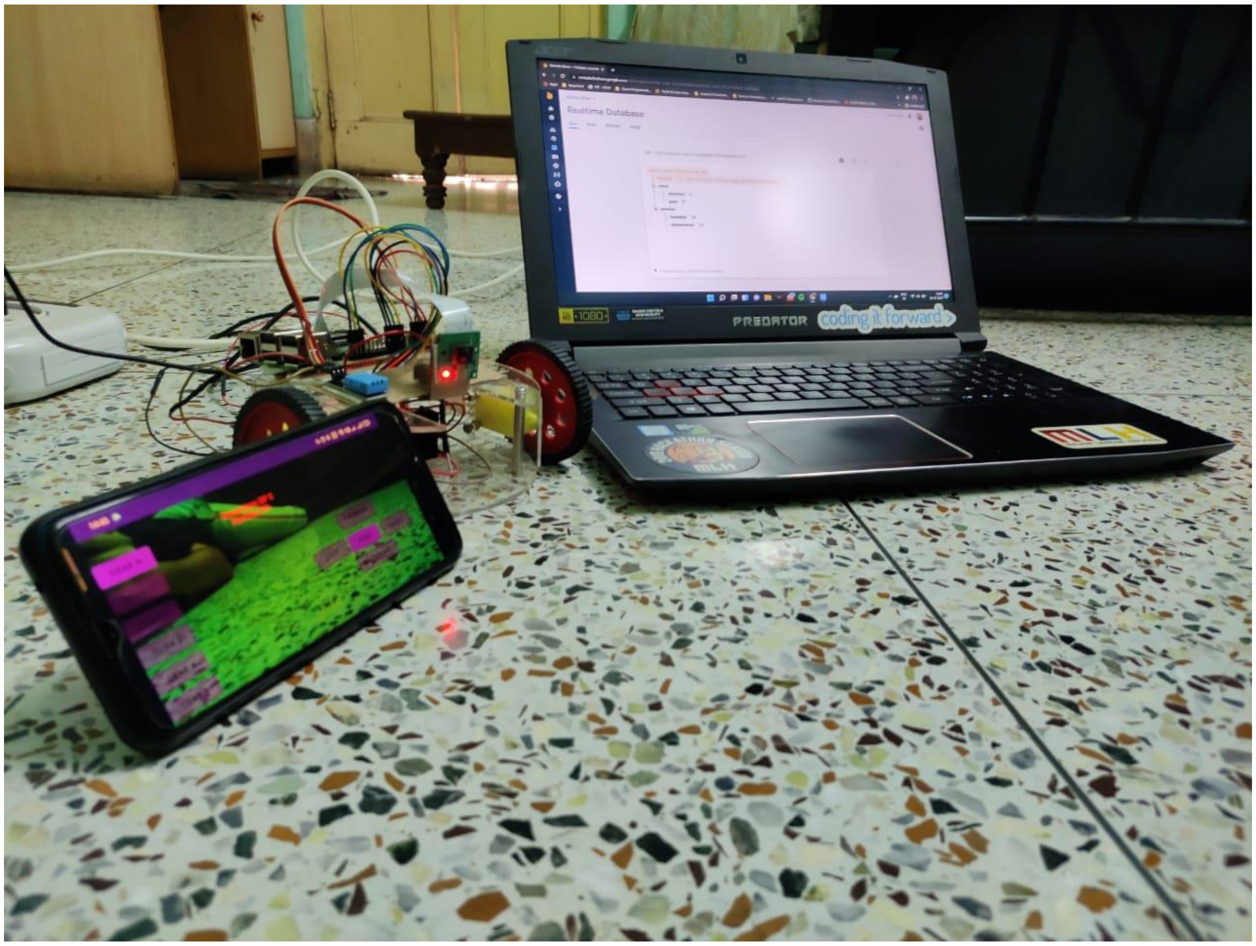
The above code snippet is of the main activity of the android app.



```
                android:text="Gear 5" />

        </LinearLayout>
        <RelativeLayout
            android:layout_width="300dp"
            android:layout_height="200dp"
            android:layout_gravity="bottom|end"
            android:layout_marginBottom="30dp"
            android:layout_marginEnd="30dp">

            <Button
                android:id="@+id/btnForward"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentTop="true"
                android:layout_alignParentEnd="true"
                android:layout_alignParentRight="true"
                android:layout_marginTop="31dp"
                android:layout_marginEnd="92dp"
                android:layout_marginRight="92dp"
                android:backgroundTint="@color/transparentViolet"
                android:gravity="center_horizontal"
                android:onClick="changeDirection"
                android:text="forward" />

            <Button
                android:id="@+id/btnBackward"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentEnd="true"
```

The above code snippet is the xml code which was used to design the UI of the android app.

The final Bot looks like this:



The above image shows the android app, the bot and the firebase real-time database on the laptop.