GESTURE RECOGNITION USING IoT

A Mini-Project Report submitted

In partial fulfillment for the Degree of

B. Tech in

Electronics and Communication Engineering

By

A.ABHIRAM **19241A0403**

N.VIGNESH TEJA **19241A0434**

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

BACHUPALLY, KUKATPALLY

HYDERABAD-500090

2021-2022

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

BACHUPALLY, KUKATPALLY HYDERABAD-500090

2021-2022

CERTIFICATE

This is to certify that this mini-project report entitled

“GESTURE RECOGNITION USING IoT”

By

A.ABHIRAM 19241A0403

N..VIGNESH TEJA 19241A0434

Submitted in partial fulfillment of the requirements for the degree of **Bachelor of Technology in Electronics and Communication Engineering** at **Gokaraju Rangaraju Institute of Engineering and Technology**, Hyderabad, during the academic year 2021-2022, is a bonafide record of work carried out under our guidance and supervision.

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

Dr. V HimaBindu

Head of Department External Examiner

DECLARATION

I declare that this mini-project report titled "*GESTURE RECOGNITION USING IoT*" submitted in partial fulfillment of the degree of **B. Tech in (Electronics and Communication)** is a record of original work carried out by me and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

A.ABHIRAM(19241A0403)

N.VIGNESH TEJA (19241A0434)

ACKNOWLEDGMENTS

There are many people who helped me directly and indirectly to complete my project successfully. I would like to take this opportunity to thank one and all.

I wish to express my sincere thanks to **Dr. V HimaBindu**, HOD, Dept. of ECE and to our principal **Dr. J Praveen** for providing the facilities to complete the dissertation.

I would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, I am very much indebted to my parents for their moral support and encouragement to achieve my goals.

Yours Sincerely

A.ABHIRAM
(19241A0403)

N.VIGNESH TEJA
(19241A0434)

TABLE OF CONTENTS

ABSTRACT

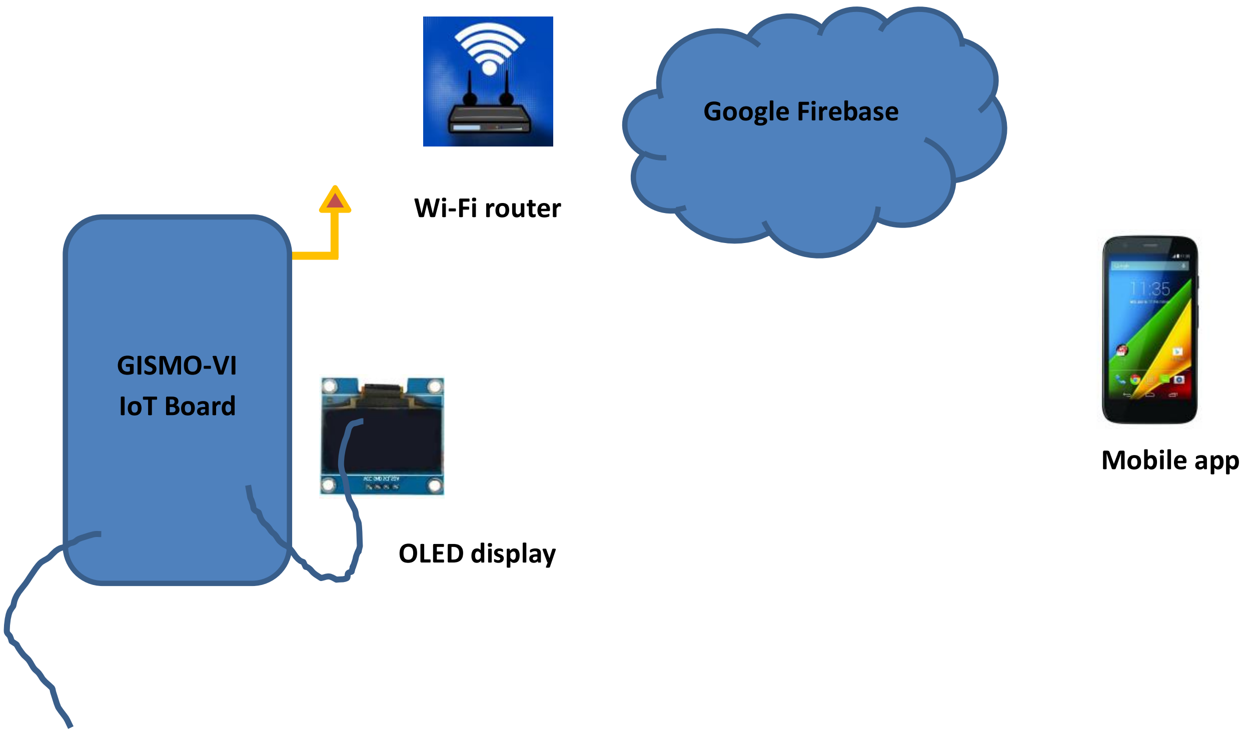
The project aims to give the accurate movement of gesture detection of a body/unit. The following gestures will be recognized:

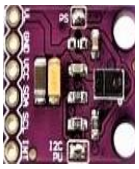
- Up to Down
- Down to Up
- Left to Right
- Right to Left
- Far to Near
- Near to Far

The project is aimed to control the machines using touchless gestures. Touchless gestures are the creative invention of human-machine interfaces. By swiping your hand over a sensor, you can control a computer, microcontroller, robot, etc.

The values will be read from APDS9960 sensor by an ESP32 controller. The ESP32 controller is a 32-bit dual core processor with Wi-Fi and Bluetooth capabilities built on-chip. The Wi-Fi capability is used to join a Wi-Fi network and connect to the Internet. The environment parameters are pushed to a cloud database – Google's Firebase which is created for the project. The database credentials – the host URL and the database authentication key are to be fed into the firmware. The Firebase database is a key-value based database and using the Firebase credentials and the key the values are accessed and displayed in the mobile app of the user.

The different components in the project are:





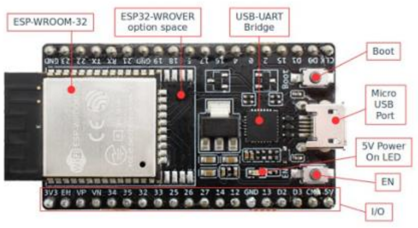
APDS9960 Sensor

Hardware:

The hardware for the project consists of:

- GISMO-VI board with:
    - ESP32 dual-core 32-bit processor with Wi-Fi and BLE
    - APDS9960 RGB and Gesture Sensor
    - 0.96" OLED display with 128x64 resolution

The ESP32 development board used is the ESP32 Dev Kit. The ESP32 board has a micro USB for programming, powering up and for transfer of data on serialline to and from the laptop. It has an on-board LED connected to GPIO2 which can be used for debugging purposes. It has a RESET and a BOOT button. The BOOT button needs to be kept pressed while downloading the program to the board. The Dev Kit also has 4 MB of external flash memory.



ESP32 Features

ESP-WROOM-32: 32-bit microprocessor

I/O pins: These pins are capable of Digital Read/Write, Analog Read/Write, PWM, IIC, SPI, DAC and much more.

Micro-USB jack: The micro USB jack is used to connect the ESP32 to our computer through a USB cable. It is used to program the ESP module as well as can be used for serial debugging as it supports serial communication

EN Button: The EN button is the reset button of the ESP module.

Boot Button: Press and upload program from Arduino IDE

Red LED: The Red LED on the board is used to indicate the power supply.

Blue LED: The Blue LED on the board is connected to the GPIO pin2.

ESP32 has Xtensa® Dual-Core 32-bit LX6 microprocessors. The ESP32 will run on breakout boards and modules from 160Mhz upto 240MHz

Internal Memory : 448 KB ROM for booting and core functions & 520 KB SRAM for data and instructions

External Flash : 4MB external Flash extendable to16MB

WiFi: ESP32 implements TCP/IP, full 802.11 b/g/n/e/i WLAN MAC protocol

Bluetooth : ESP32 supports the latest BLE Bluetooth 4.2, as well as classic Bluetooth

ESP32 Peripherals

- GPIOs
- Timers and Watchdog
- Real Time Clock
- ADC and built-in Sensors
- Digital to Analog Convertor (DAC)
- Touch Sensor
- Ultra Low Power(ULP) Co-processor
- Ethernet MAC Interface
- Universal Asynchronous Receiver Transmitter (UART)
- I2C Interface
- I2S Interface
- SPI Interface
- Pulse Width Modulation (PWM)

APDS9960

The **APDS-9960** is a multipurpose sensor that can be used for:

- **Gesture Detection**
- **Ambient Light**
- **RGB Sensing**
- **Proximity Sensing**

Gesture detection utilizes four directional photodiodes to sense reflected IR energy (sourced by the integrated LED) to convert physical motion information (i.e. velocity, direction and distance) to a digital information.

The gesture engine accommodates a wide range of mobile device gesturing requirements: simple UP-DOWN-RIGHT-LEFT gestures or more complex gestures can be accurately sensed.

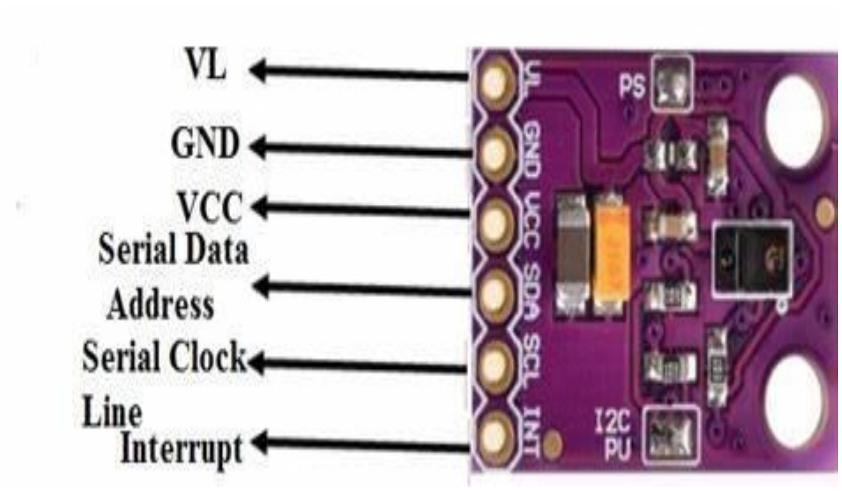


Fig: APDS9960 Gesture sensor

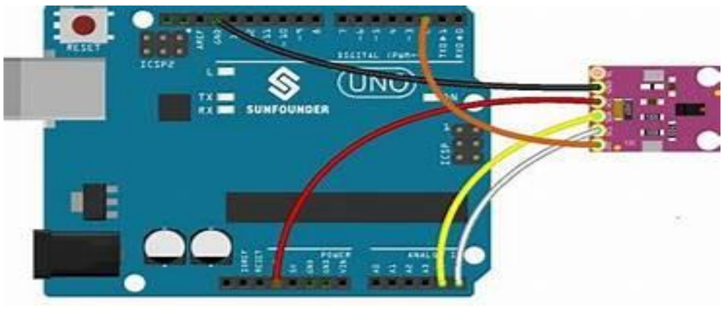


Fig: GISMO-VI board with APDS9960 Sensor

Hardware Interface with ESP32

APDS9960 pin	ESP32 pin
VL	NC
GND	GND
VCC	3.3V
Serial Data	SDA
Serial Clock	SCL
Interrupt	GPIO23

The APDS-9960 can be used to detect when objects move within range of its sensor. This shows how to throw an interrupt/gesture whenever an object moves within a certain range of the sensor. You can also change the limits to generate an interrupt/gesture whenever an object moves outside of a certain range.

OLED Display

This 0.96″ OLED Display Module offers 128×64 pixel resolution. They are featuring much less thickness than LCD displays **with good brightness and also produce better and true colors.**
This OLED Display is very compact and will serve as a local display for the project. The connection of this display with Arduino is made through the I2C(also called as IIC) serial or SPI interface.
The 0.96″ OLED Display Module **produces blue text on black background with very good contrast when supplied with DC 2.8V supply. The OLED Display Modules also offers a very wide viewing angle of about greater than 160°.**

Features

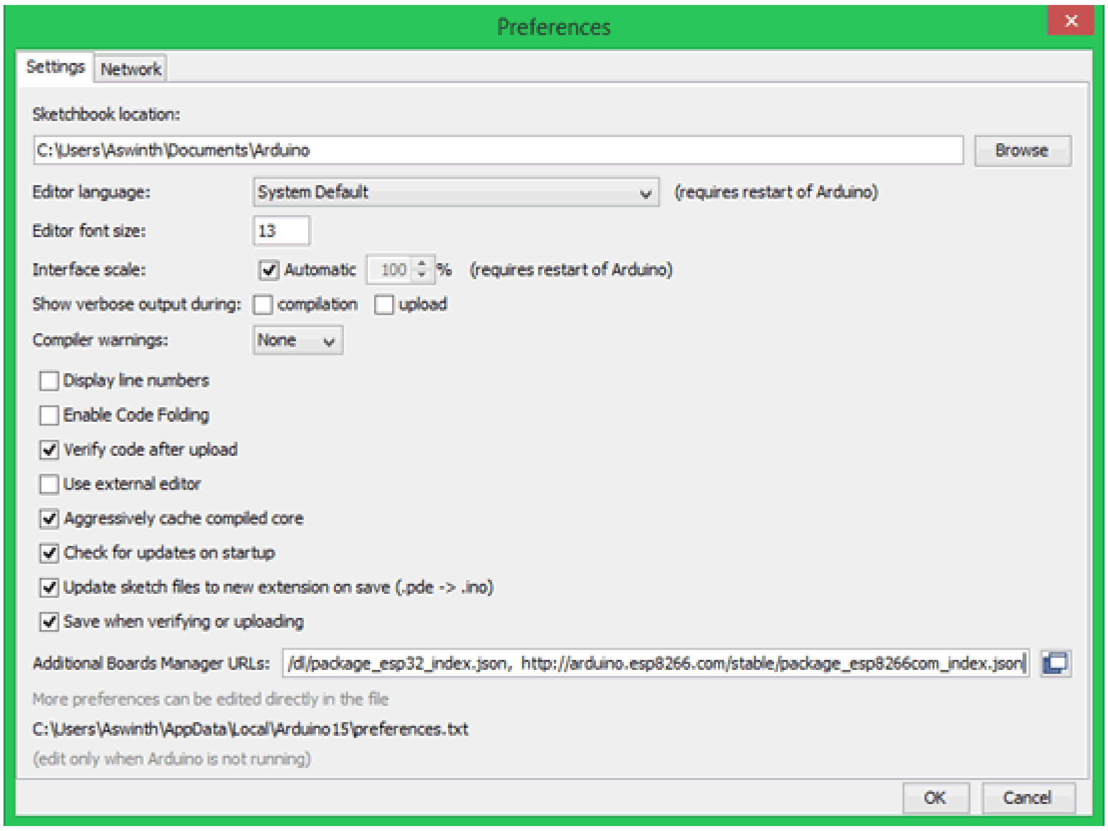
1. Driver IC: SSD1306
2. Resolution: 128 x 64
3. Visual Angle: >160°
4. Input Voltage: 3.3V ~ 6V
5. Only Need 2 I/O Port Control
6. Pixel Color: Blue

Firmware

Making Arduino IDE ready to program ESP32

The Arduino IDE is used to develop the firmware. The latest version of the Arduino IDE is loaded on the laptop from the official Arduino site. The downloaded Arduino package will not have support for the ESP32 boards. To add support for the ESP32 boards the following procedure is adopted:

1. Open the File -> Preferences -> Additional Boards URL window in the Arduino IDE
2. Copy the following string to the window:
   https://dl.espressif.com/dl/package_esp32_index.json
3. Open the Tools -> Boards -> Boards Manager and search for the ESP32 boards
4. Install the esp32 By Espressif boards
5. After downloading the relevant files from the Internet, the following group of boards will appear under boards: Tools -> Board -> Boards Manager -> ESP32 Arduino
6. A number of supported boards will appear in the ESP32 Arduino group of boards. Select ESOP32 Dev Module from the boards available under ESP32 Arduino group


Preferences

Settings | Network

Sketchbook location: C:\Users\Aswinth\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 13

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

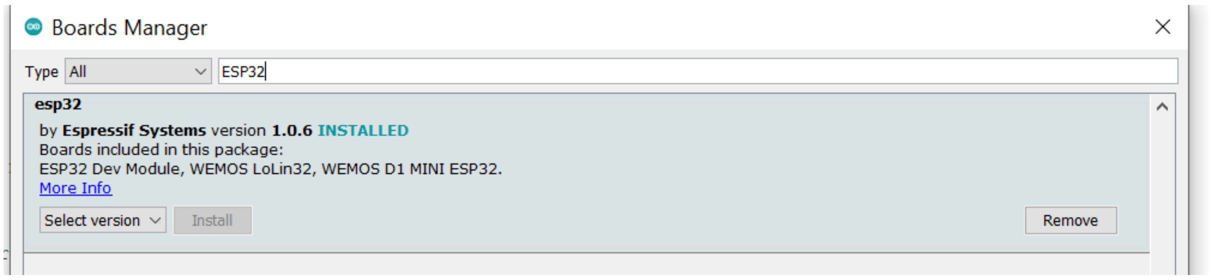
- [ ] Display line numbers
- [ ] Enable Code Folding
- [✓] Verify code after upload
- [ ] Use external editor
- [✓] Aggressively cache compiled core
- [✓] Check for updates on startup
- [✓] Update sketch files to new extension on save (.pde -> .ino)
- [✓] Save when verifying or uploading

Additional Boards Manager URLs: /dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

C:\Users\Aswinth\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)


Boards Manager

Type All | ESP32

esp32

by **Espressif Systems** version **1.0.6** **INSTALLED**

Boards included in this package:

ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.

More Info

Firmware Blocks

The firmware can be divided into the following blocks:

- Sensor interface
- OLED display interface
- Internet connectivity
- Cloud database interface

Sensor interface:

The three libraries used for interfacing the APDS9960 sensor board are:

- I2CDev.h
- APDS9960.h
- Wire.h

The APDS9960 library provides a class called APDS9960. An instance of the class is created. The class supports the following methods:

- initialize(): Takes no parameters. Is used to initialize the sensor module
- The ArduinoAPDS9960 library allows you to use the APDS9960 sensor available on the Arduino Nano 33 BLE Sense to read gestures, color, light intensity and proximity.

OLED display interface:

The two libraries used for the OLED display interface are:

- Wire,h : For basic I2C interface
- SSD1306.h: For display specific functions

The Wire.h library is natively supported by Arduino and need not be downloaded

The SSD1306 library to be installed is:

ESP8266 and ESP32 OLED Driver for SSD1306
By Thingpulse

The SSD1306 library provides a SSD1306 class. An object belonging to that class is created with the following initialization parameters:

- 7-bit I2C address of the display (0x3c)
- SCL pin of microcontroller (22)
- SDA pin of microcontroller (21)

The functions supported by the SSD1306 class are:

- init(): For initialization
- setFont(); Takes the font as a parameter. The font will decide the size of the text that will be displayed. The fonts supported are:
  - ArialMT_Plain_10
  - ArialMT_Plain_16
  - ArialMT_Plain_24
- clear(): For clearing the display
- drawSring(0,0,"fgfhgfhfhfh"): Takes three parameters – the x, y co-ordinates of the strat of the string and the string to be displayed
- display(): No parameters. The memory buffer is transferred to display

Internet connectivity

The in-built WiFi object is used to connect to the WiFi network. The WiFi object has the following functions:

- begin(): Takes two parameters:
  - SSID of the WiFi network to which we want to connect
  - Password of the WiFi network
- status(): Tells us whether the ESP32 is connected to the WiFi network or not.

- localIP(): The IP address dynamically assigned to the ESP32 by the access point (router)

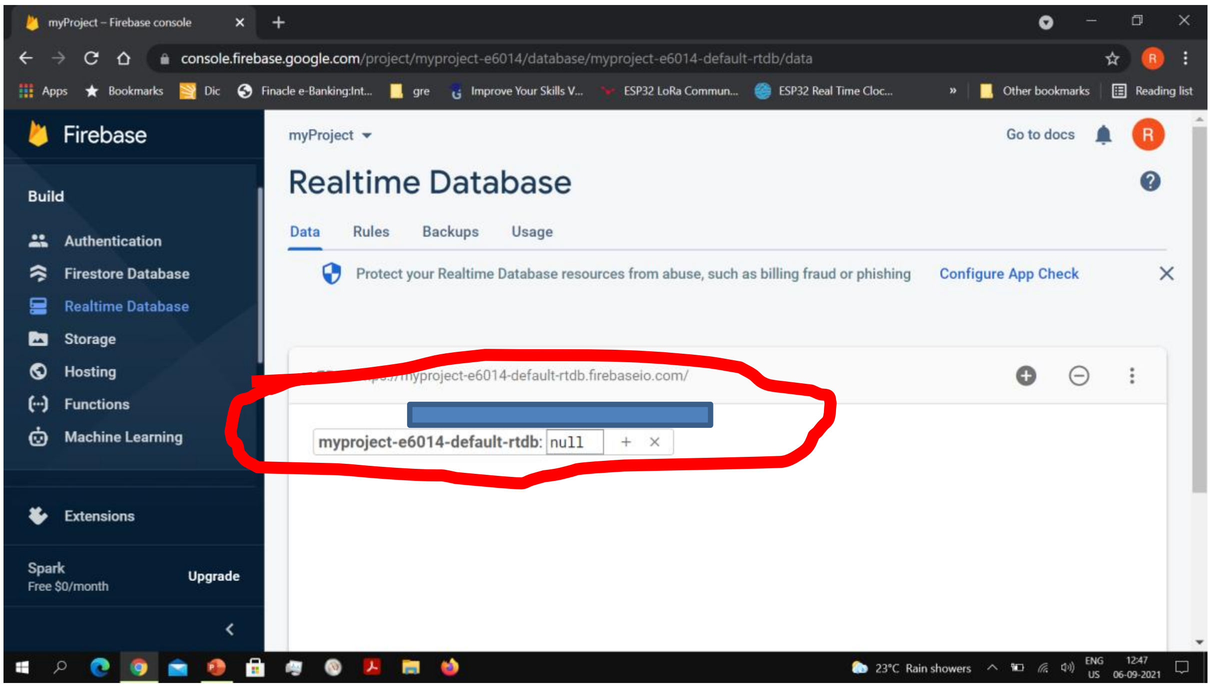
The connection to the WiFi network is not instantaneous. As long as the ESP32 is trying to connect to the WiFi network, the blue user LED will be blinking and a series of dots will appear in the Serial Monitor.Once the ESP32 is connected, an IP address will be assigned to ESP32 and the blue LED blinking will stop. This functionality is built into the WiFi_Init() function.

Cloud database access

The cloud database used in the project is Google's Firebase. It is a No-SQL database in which data is stored as Key-Value pairs. Login can be done using your Gmail ID.

The following are the credentials of the database required for access:

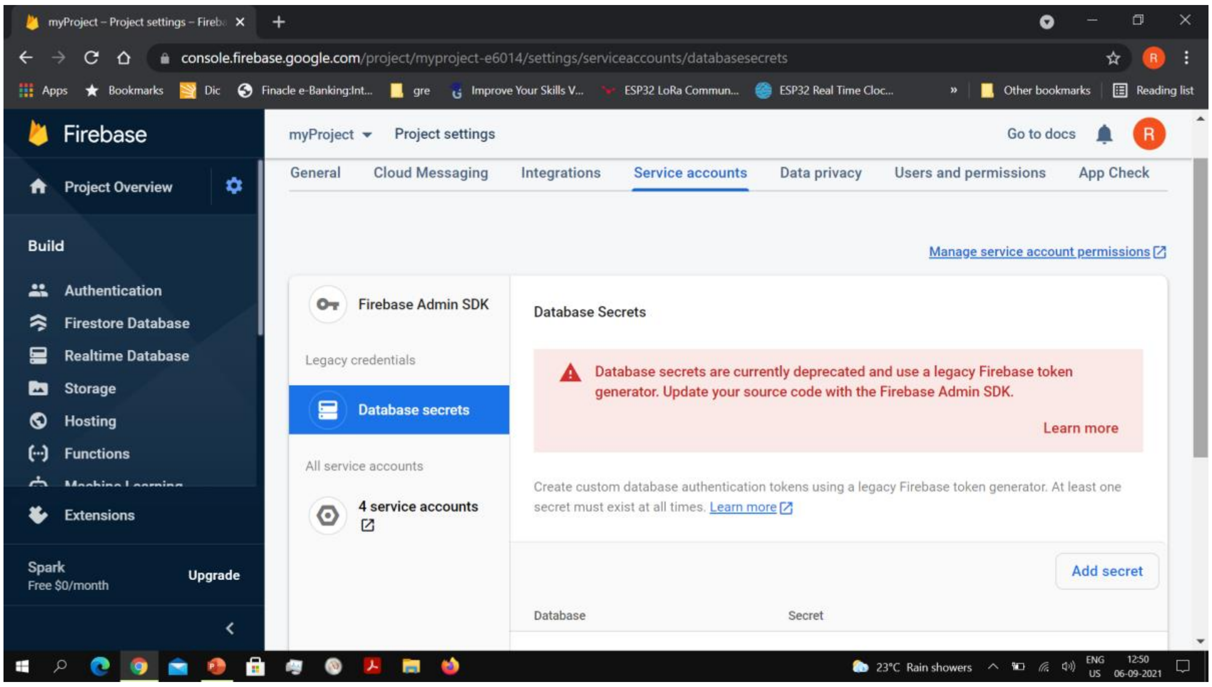
- Host URL
- Database authentication key



https://myproject-e6014-default-rtdb.firebaseio.com/

[redacted]

myproject-e6014-default-rtdb: null



Database secrets are currently deprecated and use a legacy Firebase token generator. Update your source code with the Firebase Admin SDK.

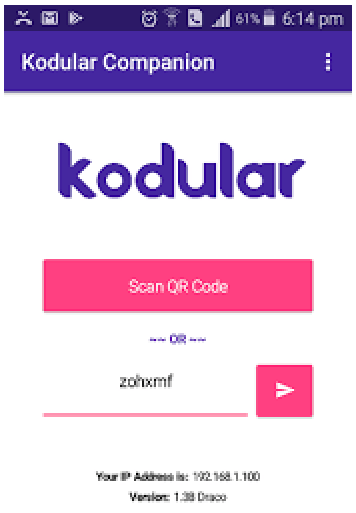
Create custom database authentication tokens using a legacy Firebase token generator. At least one secret must exist at all times. Learn more

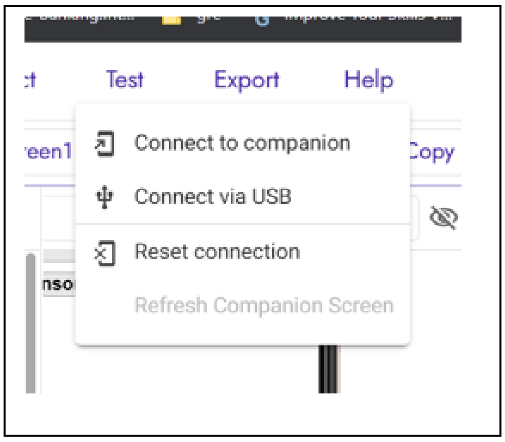
These credentials will be put in the ESP32 firmware to access the Google Firebase. The following Arduino library needs to be installed in order to gain access to the Firebase database :

Firebase ESP32 Client
By Mobizt

Mobile App development

The Kodular rapid mobile app development utility is used for mobile app development. Kodular follows a drag and drop approach for app development and there is no need to write any code. The utility as of now only supports the Android OS – so apps developed using the Kodular framework will work only for Android phones. For dynamic testing of the mobile app while it is under development, the Kodular Companion app is provided. This needs to be downloaded and installed on your mobile phone. For the Kodular app to be transferred from the laptop to the mobile phone both the devices will need to be connected to the same WiFi network. Under the Test tab in Kodular, there is a Connect to Companion option, which will open up a 2D QR code. This will be scanned by the Kodular Companion app





The utility has two modes:

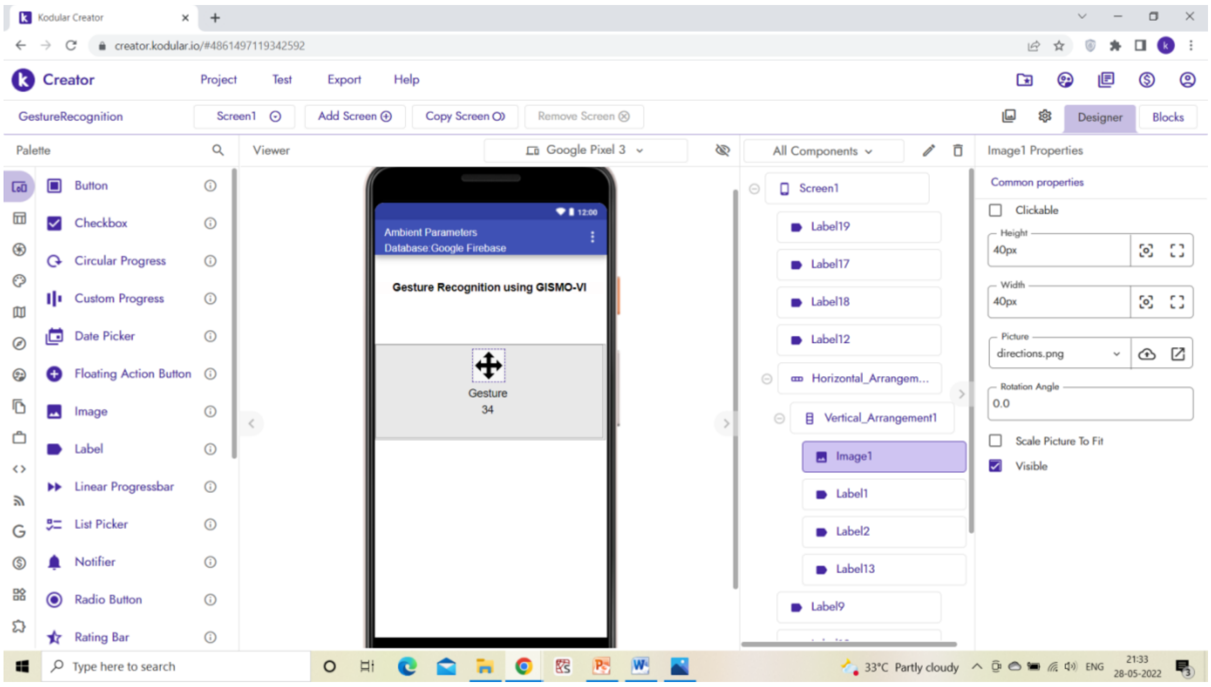
- Developer mode
- Blocks mode

The Developer mode will help define the look and feel of the mobile app. In the Developer mode, there are different functional components available – User Interface, Sensors, Connectivity, Firebase, and so on.

The User Interface developed for the mobile app has the following components:

- ✓ NEAR
- ✓ FAR
- ✓ RIGHT
- ✓ LEFT
- ✓ UP
- ✓ DOWN

The Clock and Firebase database components are non-visible components



Clickable
Height: 40px
Width: 40px
Picture: directions.png
Rotation Angle: 0.0
Scale Picture To Fit
Visible ✓

In the Blocks mode, the timer with timing set to 2 seconds will fetch the Gesture detection status (up/down/near/far/left/right) values from the Firebase database and fill it in the appropriate place in the UI. To access the Firebase datase its credentials will be put in the Firebase component