**Fall Detection using IoT**

**Description**

The projects aims to detect the position of a person carrying the unit. The unit will detect if there is a fall and will detect and report the following positions:

* Upright
* Fall
  + Forward
  + Backward
  + Left
  + Right

The project is aimed as an assistance tool for geriatric patients, so that their caretakers can continuously monitor their status. The sensor used is MPU6050 which provides :

* Acceleration in X, Y, Z axes
* Gyrometer readings in X, Y, Z axes

The project uses the accelerometer values only. The values will be read from MPU6050 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:

**Google Firebase**



**Mobile app**

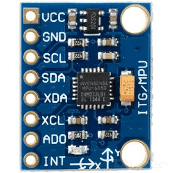


**OLED display**



**GISMO-VI IoT Board**

**Wi-Fi router**



**MPU6050 sensor**

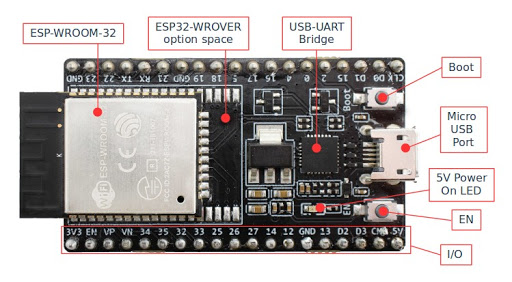
**sensor**

**Hardware**

The hardware for the project consists of:

* GISMO-VI board with:
  + ESP32 dual-core 32-bit processor with Wi-Fi and BLE
  + MPU6050 3-axes accelerometer and gyro sensor module
  + 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**

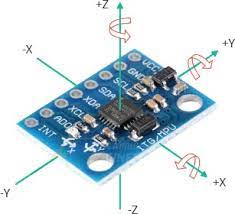
* GPIOs
* Timers and Watchdog
* Real Time Clock
* ADC and built-in Sensors
* Digital to Analog Convertor (DAC)
* Touch Sensor
* Ultra Low Power(ULP) Co-processor
* Ethernet MAC Interface
* Universal Asynchronous Receiver Transmitter (UART)
* I2C Interface
* I2S Interface
* SPI Interface
* Pulse Width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

**MPU6050**

MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package. Also, it has additional feature of on-chip Temperature sensor. It has I2C bus interface to communicate with the microcontrollers.

* MEMS 3-aixs accelerometer and 3-axis gyroscope values combined.
* Power Supply: 3-5V.
* Communication : I2C protocol.
* Built-in 16-bit ADC provides high accuracy.

IOT Lab, ECE Department

**Hardware Interface with ESP32**

|  |  |
| --- | --- |
| **MPU6050 pin** | **ESP32 pin** |
| VCC | 3.3V |
| GND | GND |
| SCL | GPIO22 |
| SDA | GPIO21 |

The MPU6050 is interfaced with the ESP32 using the I2C interface. The I2C interface has two signals – SCL and SDA. These pins on the MPU6050 are connected to I2C pins on ESP32 – GPIO22 and GPIO21. The MPU6050 works on 3.3V supply

**Data output**

**Sensitivity** : +/- 2g

**Output** : 16 bit

+/-2g corresponds to +/-32767

1g corresponds to 32767/2 = 16384

Dividing the accelerometer value by 16384 will give the value in **g**

**OLED Display**



**This 0.96″ OLED Display Module offers 128×64 pixel resolution. They are featuring much less thickness than**[LCD displays](https://robu.in/product-category/led-lcd-and-display-boards/lcd-display/)**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](https://robu.in/product-category/arduino-2/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**

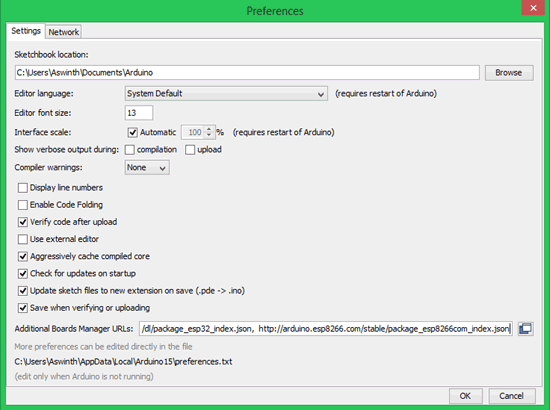
**Making Arduino IDE ready to program ESP32**

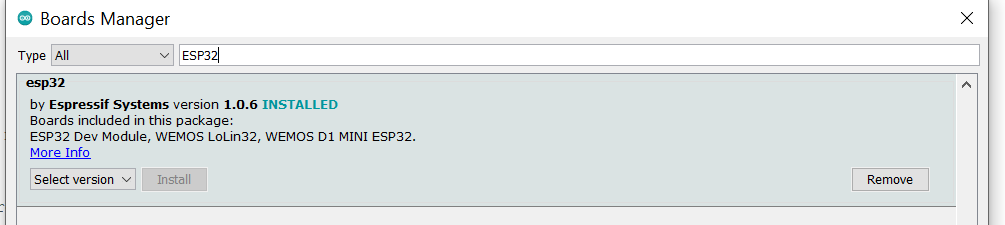
The Arduino IDE is used to develop the firmware. The latest version of the Arduino IDE is loaded on the laptop from the official Arduino site. The downloaded Arduino package will not have support for the ESP32 boards. To add support for the ESP32 boards the following procedure is adopted:

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>

1. Open the Tools -> Boards -> Boards Manager and search for the ESP32 boards
2. Install the esp32 By Espressif boards
3. After downloading the relevant files from the Internet, the following group of boards will appear under boards: Tools -> Board -> Boards Manager -> ESP32 Arduino
4. 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





**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 MPU6050 sensor board are:

* I2CDev.h
* MPU6050.h
* Wire.h

The MPU6050 library provides a class called MPU6050. An instance of the class is created. The class supports the following methods:

* initialize(): Takes no parameters. Is used to initialize the sensor module
* getMotion6(); Takes 6 parameters viz addresses of the following variables:
  + X, Y, Z accelerometer variables
  + X, Y, Z gyrometer variables
  + Only the accelerometer values are used. The values are signed 16-bit values. The sensitivity is +/- 2G
  + +/-32767 corresponds to +/- 2G
  + The 16-bit values are divided by 16384 to get the acceleration in G units

**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)

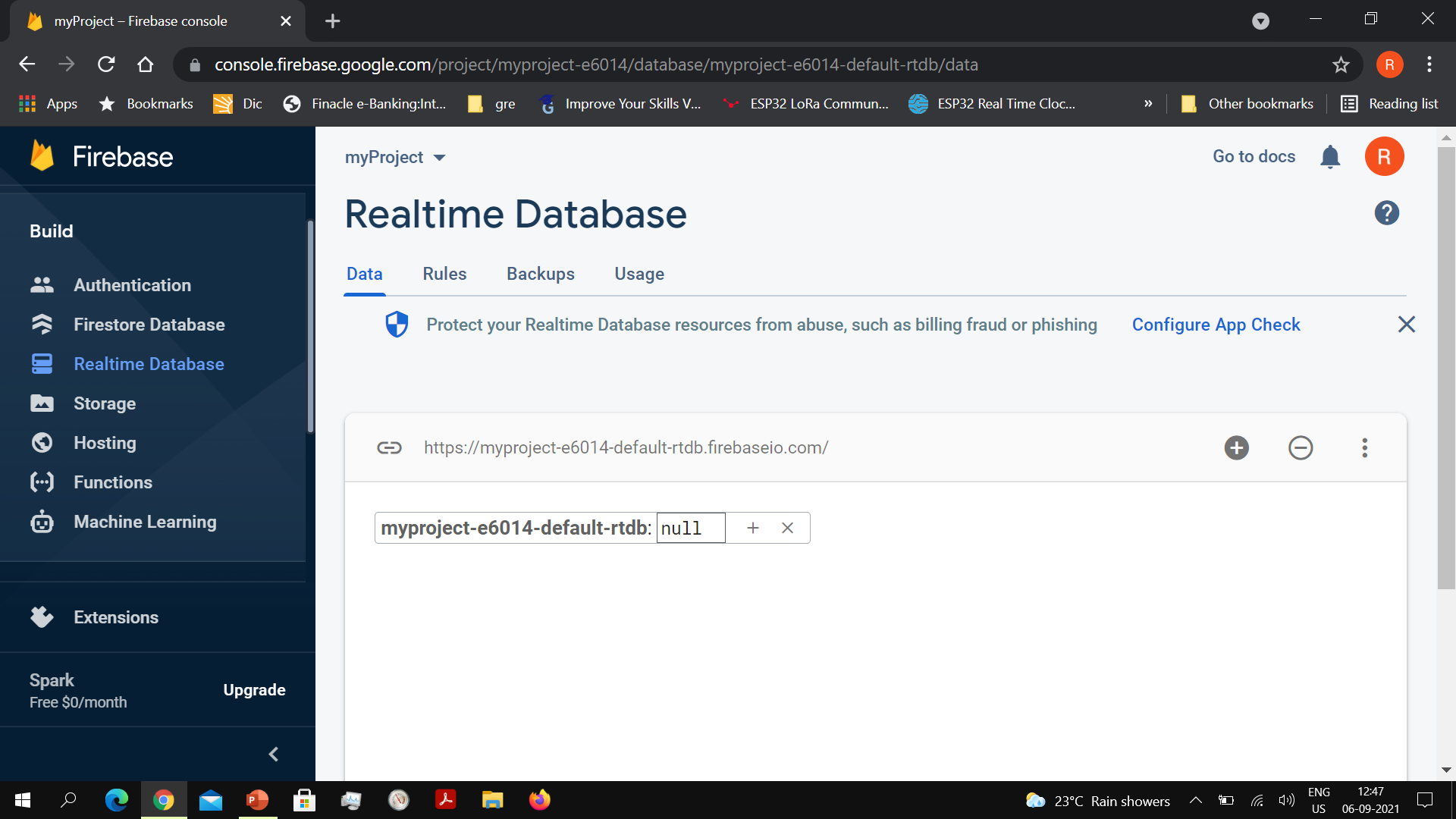
The connection to the WiFi network is not instantaneous. As long as the ESP32 is trying to connect to the WiFi network, the blue user LED will be blinking and a series of dots will appear in the Serial Monitor.Once the ESP32 is connected, an IP address will be assigned to ESP32 and the blue LED blinking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

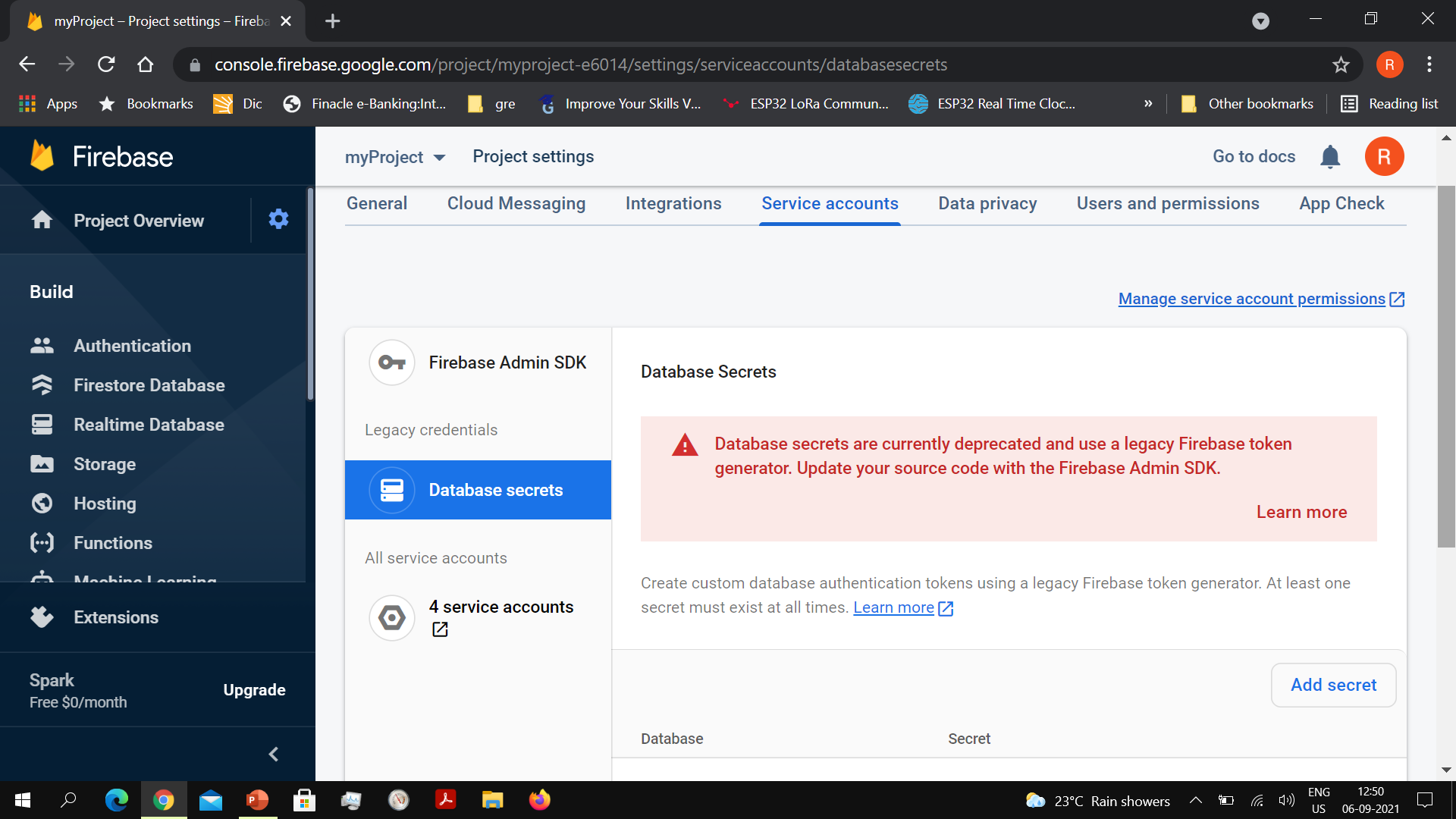
**Cloud database access**

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





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

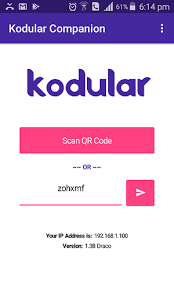
The keys used to store the parameter values in the project are:

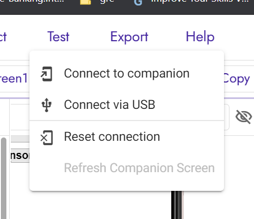
* IOTLAB/Fall\_Detector/Limit
* IOTLAB/ Fall\_detector /Position

The limit for detecting a fall can be set from the mobile app in terms of G nad the value will be stored in the Firbase database. This limit value will be read by ESP32 and will be used to detect a fall.

**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:

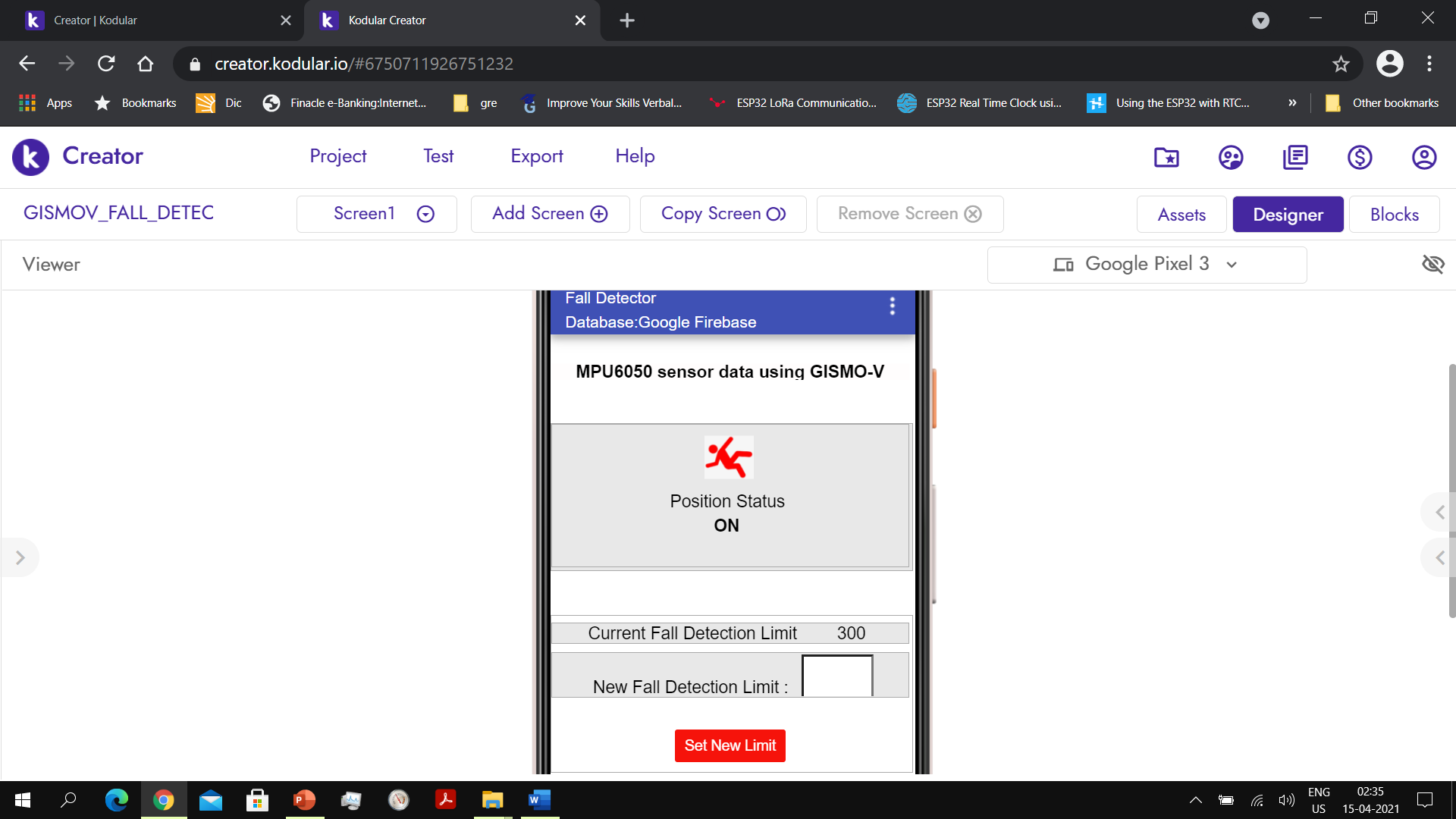
* Developer mode
* Blocks mode

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:

* Position Status display
* Current Fall detector Limit
* Facility for entering and setting a new limit
* A clock timer
* A Firebase database component

The Clock and Firebase database components are non-visible components



In the Blocks mode, the timer with timing set to 2 seconds will fetch the fall detection status (upright/forward/backward/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