Gokaraju Rangaraju
Institute of Engineering and Technology

Internet of Things Laboratory
Record of Experiments

Name:			
Reg. No.:			
Branch:			
Class:		**Section**	

GRIET - ECE
Focused Engineering

CERTIFICATE

This is to certify that it is a bonafide record of practical work done by Mr/Ms ____________________ bearing the Roll number ___________________in the Internet of things Laboratory in__________semester of_____year during 20____20__ ______.

LAB In-Charge

Gokaraju Rangaraju Institute of Engineering and Technology

Department of Electronics and Communication Engineering

Institute Vision and Mission

Vision

To be among the best of the institutions for engineers and technologists with attitudes, skills and knowledge and to become an epicentre of creative solutions.

Mission

To achieve and impart quality education with an emphasis on practical skills and social relevance.

Department of ECE, GRIET

Vision

"To be recognized globally through quality education for well-qualified engineers, innovative in research, ethically strong, entrepreneurial with good managerial skills and successful in their professional careers."

Mission

"To create knowledgeable and skilled persons with strong fundamentals, through rigorous curriculum that develops ability to solve problems individually and in teams."

"To provide professional leaders to society with research attitudes in the core areas of Electronics and Communication with collaborations to impact quality of academics, Industry and social needs."

"To create competent manpower with deep awareness on ethics, skills and leadership qualities to our profession and society."

Program Educational Objectives (PEOs)

PEO1: Graduates will shoulder responsibilities in their chosen careers including supportive and roles in multidisciplinary teams.

PEO2: Graduates will effectively understand, use and develop modern data storage, interpretation, analytical and simulation technologies and other engineering concepts and tools in the field of electronics and communications.

PEO3: Graduates will communicate effectively, recognize and incorporate societal needs and constraints in their professional endeavours, and practice their profession with high regard to legal and ethical responsibilities.

PEO4: Graduates will be able to engage in life-long learning and be abreast with the changing technologies in their profession and to be leaders in the society.

Gokaraju Rangaraju Institute of Engineering and Technology

Department of Electronics and Communication Engineering

Program Outcomes B.Tech (ECE)

The graduate will have:

PO1. **Engineering knowledge:** Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Comprehend and analyze real-time engineering problems of communication, signal processing, VLSI and Embedded systems by applying the fundamental concepts of Electronics.

PSO2: Demonstrate proficiency in utilization of modern hardware and software tools along with analytical implementation skills to arrive at appropriate solutions.

Gokaraju Rangaraju Institute of Engineering and Technology

Department of Electronics and Communication Engineering

COURSE OBJECTIVES AND COURSE OUTCOMES

Academic Year : 2023-2024 Year/ Semester: III/ I

Name of the Course: Internet of Things Laboratory

Course Code: GR20A3040 L/T/P/C: 0/0/3/1.5

III Year I Semester

Course Objectives:

1. To gain knowledge of various sensors and actuators used in implementation of IoT modules.
2. To understand various techniques and algorithms used for interfacing the sensors with the microcontrollers.
3. To access the Cloud via gateway and enable information transition to and from the Cloud.
4. To design a complete IoT ecosystem incorporating different communication protocols.
5. To build Mobile applications for implementing various dataflow scenarios.

Course Outcomes:

1. Understand the different blocks involved in an IOT eco-system and
2. Understand interfacing techniques to connect different sensors to a microcontroller.
3. Understand how a gateway module works as a bridge between two networks.
4. Understand different communication protocols used in IOT such as HTTP and web sockets.
5. Understand the programming of mobile applications to push and pull data from the cloud. and apply the concepts to implement a complete IOT eco-system with different dataflow scenarios.

Gokaraju Rangaraju Institute of Engineering and Technology

Department of Electronics and Communication Engineering

Course Articulation Matrix:

Course Outcomes	POs												PSOs	
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1		3			3								3	
CO2	3		3	3									3	2
CO3	2	2		2	3								2	
CO4	3	3		3										3
CO5	3	3		3	3	3			2	2	2	2	3	3

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

Course Code: GR20A3040 **L/T/P/C: 0/0/3/1.5**
III Year I Semester

Task-1: Microcontroller – Sensor/ Actuator Interfacing

Programming a Generic Sensor Board to interface the following sensors/actuators

1. Blinking of LED
2. Buzzer Tone
3. Relay control for switching applications
4. Ultrasonic sensor module
5. Soil Moisture Sensor Module
6. MEMS Sensor Module (INMP 441)
7. PIR Sensor Module
8. Gesture Sensor Module
9. Environment Monitoring Module (BMP 280)
10. Heart Rate Monitoring Module
11. Multi-Axis Accelerometer Sensor Module
12. Magnetic switch
13. OLED Display interface
14. IR sensor Interface

Task-2: Mobile App development

Mobile app development using MIT's App Inventor and Kodular platforms to

1. Develop apps with simple UI
2. Mobile apps to push pull data from the cloud database
3. Mobile apps to push actuator commands to the cloud database

Task-3: IOT projects

Integrating different blocks to do the following IOT projects

1. Home Security System with IoT interface
2. Smart Garden with IoT interface
3. IR Remote based motor Control with IoT Interface
4. IoT based Remote Range Meter
5. Fall Detection using IoT
6. Gesture Recognition using IoT
7. Heart Rate Monitoring using IoT
8. Wake Sound Detection and Alarm Notification using IOT
9. IoT based Real Time Appliance Control
10. Environment Monitoring using IoT

Text/ Reference Books:

1. Building Arduino Projects for the Internet of Things by Adeel Javed, Apress, 2016

1. Introduction to GISMO VI Board

GISMO VI Board is prepared by the Faculty of ECE, Department of Electronics GRIET.

The main features of GISMO VI board are:

It comprises of ESP32 loaded with lots of new features when compared with its predecessor. It combines WIFI and Bluetooth capabilities and dual-core. It has Wireless connectivity WIFI**:** 150.0 Mbps data rate with HT40. The processor used is Dual-Core 32-bit LX6 microprocessor, running at 160 or 240 MHz The ROM size is 448KB and SRAM size is 520KB.

The ESP-32 is having the following features:

- peripheral interface with DMA that includes capacitive touch
- ADCs (Analog-to-Digital Converter)
- DACs (Digital-to-Analog Converter)
- I²C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- SPI (Serial Peripheral Interface)
- I²S (Integrated Interchip Sound)
- RMII (Reduced Media-Independent Interface)
- PWM (Pulse-Width Modulation).
- **Security:** hardware accelerators for AES and SSL/TLS
- **Arduino IDE compatible:** you can program the ESP32 with the Arduino IDE using the Arduino core. (Windows, Mac OS X, and Linux installation instructions). You can also use other IDEs to program the ESP32 with the Arduino core (like VS Code with the Platform IO extension, for example).
- **Compatible with Micro Python**: you can program the ESP32 with Micro Python firmware

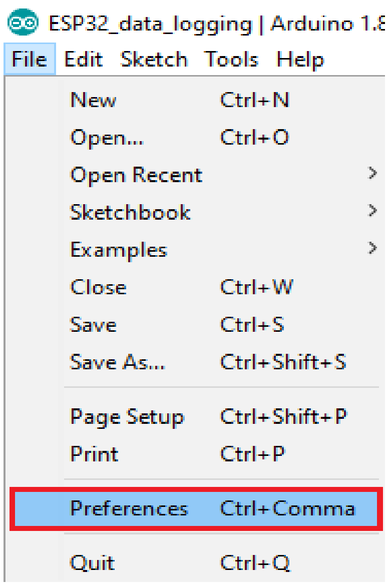
GRIET's IOT board GISMO has the following resources:

- Arduino Nano
- ESP-01 WIFI module
- DHT11 Temperature & RH sensor
- LDR
- Relay
- MAX485 for RS485 communication
- XBee base
- On-board 3.3V generation
- A1-A7 pins extended out to header

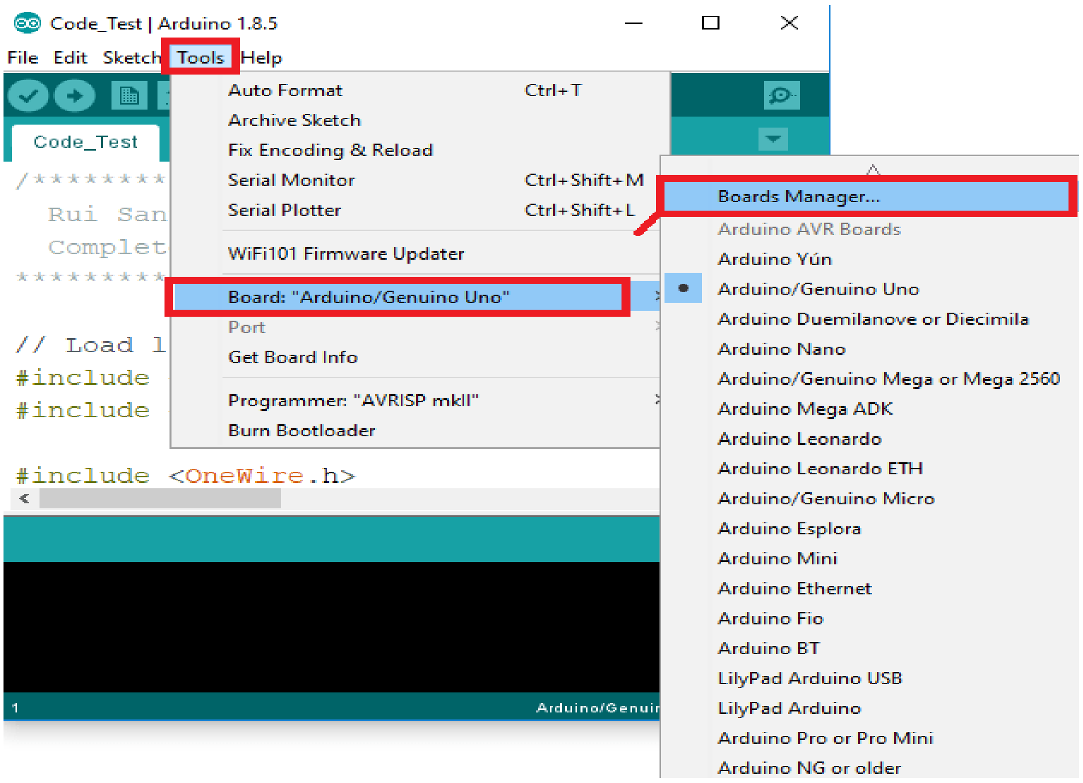
Installing ESP32 Add-on in Arduino IDE

To install the ESP32 board in your Arduino IDE, follow these next instructions:

1. In your Arduino IDE, go to **File**> **Preferences**

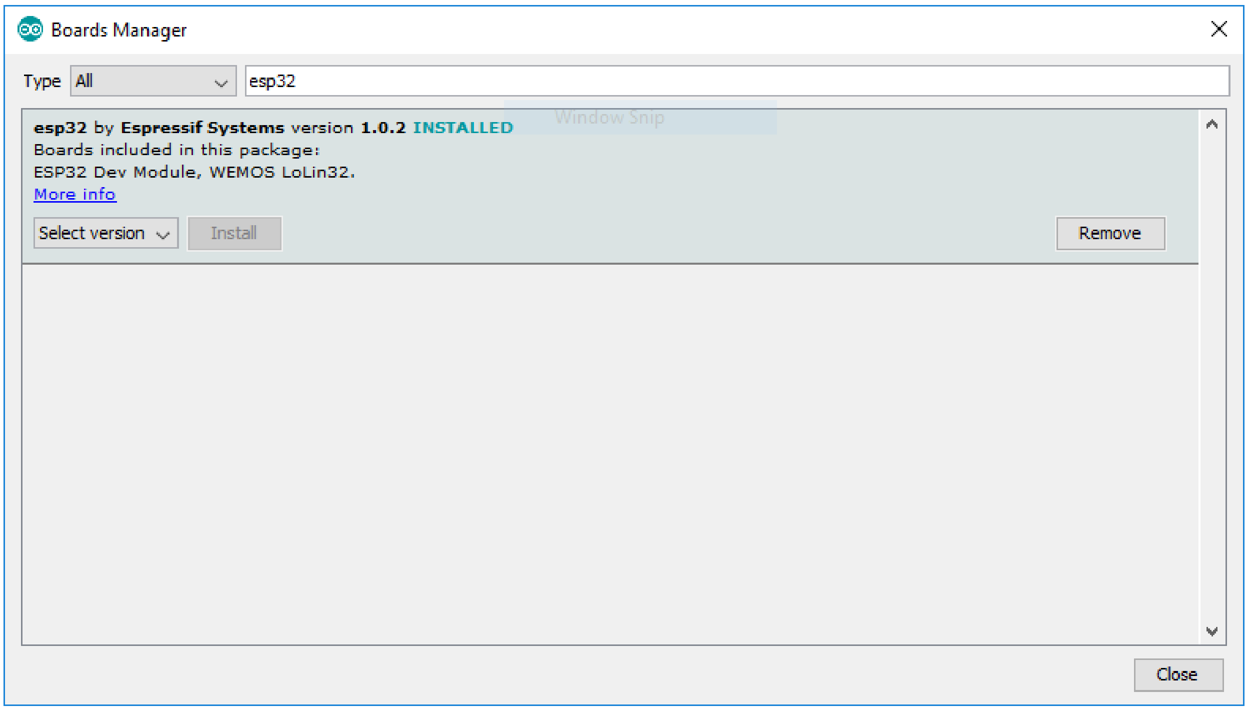


Note: if you already have the ESP8266 boards URL, you can separate the URLs with a comma as follows:
http://arduino.esp8266.com/stable/package_esp8266com_index.json



```
/********
  Rui San
  Complet
********

// Load l
#include
#include

#include <OneWire.h>
```

That’s it. It should be installed after a few seconds.



esp32 by Espressif Systems version 1.0.2 INSTALLED
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.

TASK1: SENSOR INTERFACING WITH ESP32 ON GISMOVI BOARD

S.NO	List of Experiments
1.	Blinking of LED
2	Buzzer Tone
3	Relay control for switching applications
4	Ultrasonic Sensor Module
5	Soil Moisture Sensor Module
6	MEMS Sensor Module
7	PIR sensor Module
8	Gesture Sensor Module
9	Environment Monitoring Module (BMP280)
10	Heart Rate Monitoring Module
11	Multi axis Accelerometer (MPU 6050)
12	Magnetic Switch Interface
13	OLED Display Interface
14	IR sensor Interface

GISMO-VI On-board pin assignments

01: Relay

Actuator	ESP32 pin
Relay (Transistor drive)	GPIO13

Headers for external sensors:

02: Magnetic switch

Sensor signal	ESP32 pin
Sensor o/p contact 1	GPIO16
Sensor o/p contact 2	GND

03: PIR module:

Sensor signal	ESP32 pin
GND	GND
OUT	GPIO33
VCC	5V

04: Soil moisture sensor

Sensor signal	ESP32 pin
VCC	3.3V
GND	GND
DO	NC
AO	GPIO34

Note: Do not short the AO and DO pins on the PCB

05: Ultrasonic sensor HC-SR04

Sensor signal	ESP32 pin
VCC	5V
Trigger	GPIO25
Echo	GPIO26
GND	GND

Note: Echo should be level translated using 5.6K and 10K potential divider

06: OLED module

Sensor signal	ESP32 pin
SDA	GPIO21
SCL	GPIO22
VCC	3.3V
GND	GND

07: APDS9960 module

Sensor signal	ESP32 pin
INT	GPIO23
SCL	GPIO22
SDA	GPIO21
VCC	3.3V
GND	GND

08: BMP280 module

Sensor signal	ESP32 pin
SDO	NC
CSB	NC

SDA	GPIO21
SCL	GPIO22
GND	GND
VCC	3.3V

09: MPU6050 module

Sensor signal	**ESP32 pin**
INT	NC
ADO	NC
XCL	NC
XDA	NC
SDA	GPIO21
SCL	GPIO22
GND	GND
VCC	3.3V

10: MAX30102 module

Sensor signal	**ESP32 pin**
VIN	3.3V
SCL	GPIO22
SDA	GPIO21
INT	NC
IRD	NC
RD	NC
GND	GND

Note: SCL and SDA lines are to be pulled up

11: IR Receiver

Sensor signal	**ESP32 pin**
OUT	GPIO4
GND	GND
VCC	3.3V

12: INMP441

Sensor signal	**ESP32 pin**
L/R	GND
WS	GPIO27
SCK	GPIO14
GND	GND
VDD	3.3V
SD	GPIO12

13: Buzzer

Actuator	**ESP32 pin**
Buzzer (Transistor drive)	GPIO32

Task 1: Interfacing with Sensors and Actuators

1. Blinking of LED

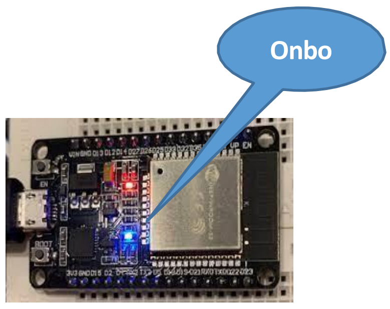
Aim: Blinking of LED for 1 second interval on GISMO VI Board.

Components Required:

- GISMO VI Board
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

LED



GISMO VI connections

Actuator	ESP32 pin
Buzzer (Transistor drive)	GPIO2

Working Principle:

A light-emitting diode is a two-lead semiconductor light source. It is a p–n junction diode that emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the colour of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor.

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code for blinking the LED.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Check the blinking of LED for 1 second on GISMO VI at GPIO Pin 2 of ESP-32.

Arduino Code:

```
void setup()
{
pinMode(2 , OUTPUT);
}
void loop()
{
digitalWrite(2 , HIGH);
delay(1000);
digitalWrite(2, LOW);
delay(1000);
}
```

Observations:

Result:

2. Buzzer Tone

Aim: Controlling the Buzzer to turn on and off for 2 seconds interval with ESP-32 Microcontroller on GISMO VI Board.

Components Required:

- GISMO VI Board
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

Buzzer **GISMO VI On board Connections**

Actuator	ESP32 pin
Buzzer (Transistor drive)	GPIO32

Working principle:

The buzzer is a sounding device that can convert audio signals into sound signals. It is usually powered by DC voltage. It is widely used in alarms, computers, printers and other electronic products as sound devices. The piezoelectric buzzer uses the piezoelectric effect of the piezoelectric ceramics and uses the pulse current to drive the vibration of the metal plate to generate sound. Piezoelectric buzzer is mainly composed of multi-resonator, piezoelectric plate, impedance matcher, resonance box, housing, etc. Some of the piezoelectric buzzers are also equipped with light-emitting diodes. The multi-resonator consists of transistors or integrated circuits. When the power supply is switched on (1.5~15V DC operating voltage), the multi-resonator oscillates and outputs 1.5~2.5kHz audio signal. The impedance matcher pushes the piezoelectric plate to generate sound.

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code for turning on the buzzer.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Check the buzzer sound for 2 seconds on GISMO VI at GPIO Pin 32 of ESP-32.

Arduino Code:

```
void setup(){
pinMode(32,OUTPUT);
}
void loop(){
digitalWrite(32,HIGH);
delay(2000);
digitalWrite(32,LOW);
delay(2000);
}
```

Observations:

Result:

3. Relay Module

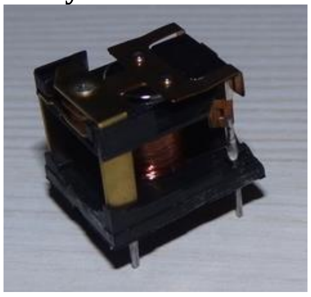
Aim: Controlling the Relay to turn on and off for 2 seconds interval with ESP-32 Microcontroller on GISMO VI Board.

Components Required:

- GISMO VI Board
- Laptop with Arduino IDE
- USB cable

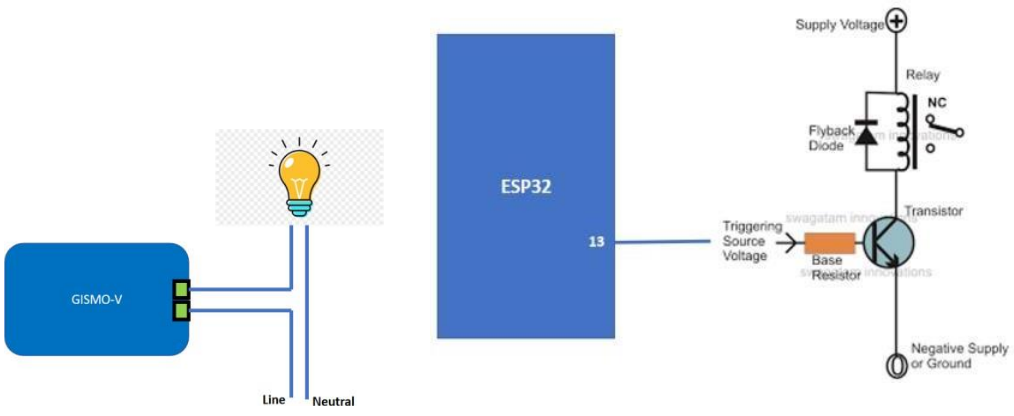
On-board pin assignments:

Relay



GISMO VI connections

Actuator	ESP32 pin
Relay (Transistor drive)	GPIO13



Working principle:

A **relay** is an electrically operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals. The switch may have any number of contacts in multiple contact forms, such as make contacts, break contacts, or combinations It works on the principle of an electromagnetic attraction. When the circuit of the relay senses the current, it energises the electromagnetic field which produces the temporary magnetic field. This magnetic field moves the relay armature for opening or closing the connections. The small power relay has only one contacts, and the high-power relay has two contacts for opening the switch.

The single pole single throw relay has pin-No contact (NC). If relay switch is on, the NO contacts will close and the Line circuit will be complete and the device to be controlled get activated. A relay can control both A.C. and D.C. circuits.

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code for controlling the Relay.
3. **Connect GISMO VI board to laptop, compile and upload code to the board.**
4. **Check the Relay Module on GISMO VI at GPIO Pin 13 of ESP-32 Microcontroller.**

Arduino Code:

```
void setup()
{
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
}

void loop()
{
  // put your main code here, to run repeatedly:
 digitalWrite(13,HIGH);
 delay(2000);
 digitalWrite(13,LOW);
 delay(2000);
}
```

Observations:

Result:

4. Ultrasonic Sensor Module (HC-SR04)

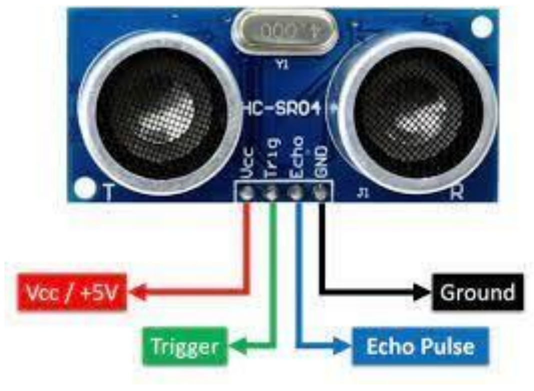
Aim: To interface ESP32 Microcontroller with Ultrasonic sensor to read the distance values.

Components Required:

- GISMO V1 Board and Ultrasonic Sensor (HC-SR04)
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

Ultrasonic Sensor



GISMOVI connections

Sensor signal	ESP32 pin
VCC	5V
Trigger	GPIO25
Echo	GPIO26
GND	GND

Working principle:

Ultrasonic sensors use electrical energy and a ceramic transducer to emit and receive mechanical energy in the form of sound waves. Sound waves are essentially pressure waves that travel through solids, liquids and gases and can be used in industrial applications to measure distance or detect the presence or absence of targets.

HCSR04 Specifications

- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working current: 15mA
- Effectual Angle: <15º
- Ranging Distance: 2400 cm
- Resolution: 0.3 cm
- Measuring Angle: 30º
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm
- Weight: approx. 10 g

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code for measuring the distance.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Measure the distance using ultrasonic sensor (**HC-SR04)** and display it on a serial monitor.

Arduino Code:

```
 // defines pins numbers
 const int trigPin = 25;
 const int echoPin = 26;
 // defines variables
 long duration;
 int distance;
 void setup() {
 pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
 pinMode(echoPin, INPUT); // Sets the echoPin as an Input
 Serial.begin(115200); // Starts the serial communication
 }
 void loop() {
 // Clears the trigPin
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 // Sets the trigPin on HIGH state for 10 micro seconds
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 // Reads the echoPin, returns the sound wave travel time in microseconds
 duration = pulseIn(echoPin, HIGH);
 // Calculating the distance
 distance= duration*0.034/2;
 // Prints the distance on the Serial Monitor
 Serial.print("Distance: ");
 Serial.println(distance);
 delay(1000);
}
```

Observations:

Result:

5. Soil Moisture Sensor Module

Aim: To interface ESP32 Microcontroller with Soil Moisture Sensor to read the Moisture values.

Components Required:

- GISMO V1 Board and Soil Moisture Sensor
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

Soil Moisture sensor

GISMO VI connections

Sensor signal	ESP32 pin
VCC	3.3V
GND	GND
DO	NC
AO	GPIO34

Working principle:

In smart garden applications, watering has to be done only when it is required, for avoiding over-watering or under-watering., such systems definitely require soil moisture. The fork-shaped probe with two exposed conductors, acts as a variable resistor (just like a potentiometer) whose resistance varies according to the water content in the soil. This resistance is inversely proportional to the soil moisture: The more water in the soil means better conductivity and will result in a lower resistance. The less water in the soil means poor conductivity and will result in a higher resistance. The sensor produces an output voltage according to the resistance, which by measuring we can determine the moisture level.

To get accurate readings out of soil moisture sensor, it is recommended that the sensor must be first calibrated based on the type of soil used.

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code for displaying the soil moisture value.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Check the soil moisture value on the serial monitor.

Arduino Code:

```
void setup() {
  // put your setup code here, to run once:
    Serial.begin(115200);
  }

void loop() {
  // put your main code here, to run repeatedly:
  int smValue = 4095 - analogRead(34);
  Serial.println(smValue);
  delay(1000);
}
```

Observations:

Result:

6. MEMS sensor Module

Aim: To interface ESP32 Microcontroller with INMP441 MEMS sensor to record sound using a MEMS microphone, transform to frequencydomain using FFT, detect presence of the specific frequency of a bell.

Components Required:

- GISMO VI Board and INMP441 MEMS sensor
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

MEMS INMO441 sensor **GISMO VI connections**

Sensor signal	ESP32 pin
L/R	GND
WS	GPIO27
SCK	GPIO14
GND	GND
VDD	3.3V
SD	GPIO12

Working principle:

The INMP441 is a high-performance, low-power, digital output, omnidirectional MEMS microphone with a bottom port. The complete INMP441 solution consists of a MEMS sensor, signal conditioning, analog to digital converter, anti-aliasing filter, power management, and industry-standard 24-bit I2S interface.

The I2S interface allows the INMP441 to be directly connected to digital processors such as DSPs and microcontrollers without the need for an audio codec for use in the system. The INMP441 has a high signal-to-noise ratio and is an excellent choice for near-field applications. The INMP441 has a flat wideband frequency response that results in a high definition of natural sound.

Interface definition

- SCK: Serial data clock for I2S interface
- WS: Serial data word selection for I2S interface
- L/R: Left/Right channel selection.
- When set to low, the microphone outputs a signal on the left channel of the I2S frame.
- When set to high level, the microphone outputs signals on the right channel
- SD: Serial data output of the I2S interface.
- VCC: Input power, 1.8V to 3.3V.

- GND: power ground

Features and Specifications:

- Digital I2S interface with high precision 24-bit data
- High signal to noise ratio is 61 dBA
- High sensitivity – 26 dBFS
- Stable frequency response from 60 Hz to 15 kHz
- Low power consumption: low current consumption 1.4 mA
- High PSR: -75 dBFS

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code to record sound using a MEMS microphone, transform to frequencydomain using FFT, detect presence of the specific frequency of a bell.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. To detect sound of specific frequency using FFT and display it on a serial monitor.

Arduino Code:

```
#include <Arduino.h>
#include <driver/i2s.h>
#include "arduinoFFT.h"

// size of noise sample
#define SAMPLES 1024

const i2s_port_t I2S_PORT = I2S_NUM_0;
const int BLOCK_SIZE = SAMPLES;

#define OCTAVES 9

// our FFT data
static float real[SAMPLES];
static float imag[SAMPLES];
static arduinoFFT fft(real, imag, SAMPLES, SAMPLES);
static float energy[OCTAVES];
// A-weighting curve from 31.5 Hz ... 8000 Hz
static const float aweighting[] = {-39.4, -26.2, -16.1, -8.6, -3.2, 0.0, 1.2, 1.0, -1.1};

static unsigned int bell = 0;
static unsigned int fireAlarm = 0;
static unsigned long ts = millis();
static unsigned long last = micros();
static unsigned int sum = 0;
static unsigned int mn = 9999;
static unsigned int mx = 0;
static unsigned int cnt = 0;
static unsigned long lastTrigger[2] = {0, 0};

static void integerToFloat(int32_t *integer, float *vReal, float *vImag, uint16_t
samples)
{
   for (uint16_t i = 0; i < samples; i++)
   {
      vReal[i] = (integer[i] >> 16) / 10.0;
      vImag[i] = 0.0;
   }
}

// calculates energy from Re and Im parts and places it back in the Re part (Im part
is zeroed)
static void calculateEnergy(float *vReal, float *vImag, uint16_t samples)
{
   for (uint16_t i = 0; i < samples; i++)
   {
      vReal[i] = sq(vReal[i]) + sq(vImag[i]);
      vImag[i] = 0.0;
   }
}

// sums up energy in bins per octave
static void sumEnergy(const float *bins, float *energies, int bin_size, int
num_octaves)
{
   // skip the first bin
   int bin = bin_size;
   for (int octave = 0; octave < num_octaves; octave++)
   {
      float sum = 0.0;
      for (int i = 0; i < bin_size; i++)
      {
         sum += real[bin++];
      }
      energies[octave] = sum;
      bin_size *= 2;
   }
}

static float decibel(float v)
{
   return 10.0 * log(v) / log(10);
}

// converts energy to logaritmic, returns A-weighted sum
static float calculateLoudness(float *energies, const float *weights, int
num_octaves, float scale)
```

```
{
   float sum = 0.0;
   for (int i = 0; i < num_octaves; i++)
   {
      float energy = scale * energies[i];
      sum += energy * pow(10, weights[i] / 10.0);
      energies[i] = decibel(energy);
   }
   return decibel(sum);
}

void setup(void)
{
   Serial.begin(115200);
   Serial.println("Configuring I2S...");
   esp_err_t err;

   // The I2S config as per the example
   const i2s_config_t i2s_config = {
      .mode = i2s_mode_t(I2S_MODE_MASTER | I2S_MODE_RX), // Receive,
not transfer
      .sample_rate = 22627,
      .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
      .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT, // for old esp-idf
versions use RIGHT
      .communication_format = i2s_comm_format_t(I2S_COMM_FORMAT_I2S |
I2S_COMM_FORMAT_I2S_MSB),
      .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1, // Interrupt level 1
      .dma_buf_count = 8,                    // number of buffers
      .dma_buf_len = BLOCK_SIZE,             // samples per buffer
      .use_apll = true};

   // The pin config as per the setup
   const i2s_pin_config_t pin_config = {
      .bck_io_num = 14,   // BCKL
      .ws_io_num = 27,    // LRCL
      .data_out_num = -1, // not used (only for speakers)
      .data_in_num = 12   // DOUTESP32-INMP441 wiring
   };

   // Configuring the I2S driver and pins.
   // This function must be called before any I2S driver read/write operations.
   err = i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
   if (err != ESP_OK)
   {
      Serial.printf("Failed installing driver: %d\n", err);
      while (true)
         ;
   }
   err = i2s_set_pin(I2S_PORT, &pin_config);
```

```
    if (err != ESP_OK)
    {
        Serial.printf("Failed setting pin: %d\n", err);
        while (true)
            ;
    }
    Serial.println("I2S driver installed.");
}

unsigned int countSetBits(unsigned int n)
{
    unsigned int count = 0;
    while (n)
    {
        count += n & 1;
        n >>= 1;
    }
    return count;
}

//detecting 2 frequencies. Set wide to true to match the previous and next bin as
well
bool detectFrequency(unsigned int *mem, unsigned int minMatch, double peak,
unsigned int bin1, unsigned int bin2, bool wide)
{
    *mem = *mem << 1;
    if (peak == bin1 || peak == bin2 || (wide && (peak == bin1 + 1 || peak == bin1 - 1
|| peak == bin2 + 1 || peak == bin2 - 1)))
    {
        *mem |= 1;
    }

    if (countSetBits(*mem) >= minMatch)
    {
        return true;
    }

    return false;
}

void sendAlarm(unsigned int index, char *topic, unsigned int timeout)
{
    // do not publish if last trigger was earlier than timeout ms
    if (abs(millis() - lastTrigger[index]) < timeout)
    {
        return;
    }

    lastTrigger[index] = millis();
    //publish to mqtt
    //publish(topic, "1");
}

void sendMetrics(char * topic, unsigned int mn, unsigned int mx, unsigned int avg)
{
    String payload = "{\"min\": ";
    payload += mn;
    payload += ", \"max\":";
    payload += mx;
    payload += ", \"value\":";
    payload += avg;
    payload += "}";

    Serial.println(payload);
    //publish to mqtt
    //publish(topic, (char *)payload.c_str());
}

void calculateMetrics(int val) {
  cnt++;
  sum += val;

  if (val > mx)
  {
      mx = val;
  }

  if (val < mn)
  {
      mn = val;
  }
}

void loop(void)
{
    if (micros() - last < 45200) {
      // send mqtt metrics every 10s while waiting and no trigger is detected
      if (millis() - ts >= 10000 && bell == 0 && fireAlarm == 0)
      {
          //Serial.println(cnt[0]);
          sendMetrics("home/noise", mn, mx, sum / cnt);
          cnt = 0;
          sum = 0;
          mn = 9999;
          mx = 0;
          ts = millis();
      }
      return;
    }
    last = micros();

    static int32_t samples[BLOCK_SIZE];

    // Read multiple samples at once and calculate the sound pressure
    size_t num_bytes_read;
    esp_err_t err = i2s_read(I2S_PORT,
                             (char *)samples,
                             BLOCK_SIZE, // the doc says bytes, but its elements.
                             &num_bytes_read,
                             portMAX_DELAY); // no timeout
    int samples_read = num_bytes_read / 8;

    // integer to float
    integerToFloat(samples, real, imag, SAMPLES);

    // apply flat top window, optimal for energy calculations
    fft.Windowing(FFT_WIN_TYP_FLT_TOP, FFT_FORWARD);
    fft.Compute(FFT_FORWARD);

    // calculate energy in each bin
    calculateEnergy(real, imag, SAMPLES);

    // sum up energy in bin for each octave
    sumEnergy(real, energy, 1, OCTAVES);

    // calculate loudness per octave + A weighted loudness
    float loudness = calculateLoudness(energy, aweighting, OCTAVES, 1.0);

    unsigned int peak = (int)floor(fft.MajorPeak());
    //Serial.println(peak);

    // detecting 1kHz and 1.5kHz 738Hz
    if (detectFrequency(&bell, 15, peak, 33, 34, true))
    {
        Serial.println("Detected bell");
        sendAlarm(0, "home/alarm/doorbell", 2000);
    }

    //detecting frequencies around 3kHz
    if (detectFrequency(&fireAlarm, 15, peak, 140, 141, true))
    {
        Serial.println("Detected fire alarm");
        sendAlarm(1, "home/alarm/fire", 10000);
    }

    calculateMetrics(loudness);
}
```

Observations:

Result:

7. PIR Sensor Module

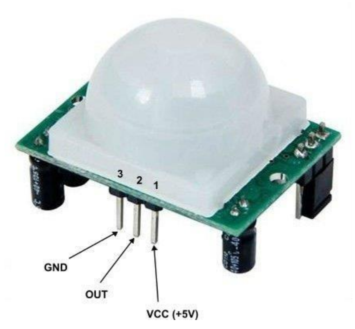
Aim: To interface ESP32 Microcontroller with PIR Sensor to detect the presence of intruders.

Components Required:

- GISMO VI Board with PIR Sensor
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

PIR sensor



GISMO VI connections

PIR sensor signal	ESP32 pin
GND	GND
OUT	GPIO33
VCC	5V

Range of PIR Sensor?

- Indoor passive infrared: Detection distances range from 25 cm to 20 m.
- Indoor curtain type: The detection distance ranges from 25 cm to 20 m.
- Outdoor passive infrared: The detection distance ranges from 10 meters to 150 meters.
- Outdoor passive infrared curtain detector: distance from 10 meters to 150 meters

Product	Measuring Range	Features
Grove PIR motion Sensor	Max Distance:3-6m(3m by default) Angle:< 120°	Measuring distance and holding time adjustable response speed:0.3s – 25s

Working principle :

This Grove – PIR Motion Sensor (Passive Infrared Sensor) can detect infrared signal caused by motion. If the PIR sensor notices the infrared energy, the motion detector is triggered and the sensor outputs HIGH on its SIG pin. The detecting range and response speed can be adjusted by 2 potentiometers soldered on its circuit board, The response speed is from 0.3s – 25s, and max 6 meters of detecting range.

This is easy to use a motion sensor with Grove compatible interface. Simply connect it to Base Shield and programming it, it can be used as a suitable motion detector for Arduino projects.

PIR sensors are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

- **PIR sensors** are small, inexpensive, low-power, easy to use and don't wear out. For that reason, they are commonly found in appliances and gadgets used in homes or businesses
- **PIR sensors** allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range.
- A **PIR sensor** is an electronic sensor that measures infrared light radiation by detecting infrared radiation (radiant heat) emitted by or reflected from objects.
- PIR sensors are mostly used in PIR-based motion detectors. Also, it used in security alarms and automatic lighting applications.
- Generally, **PIR sensor** can detect animal/human movement in a requirement range. PIR is made of a pyroelectric sensor, which can detect different levels of infrared radiation. The detector itself does not emit any energy but passively receives it.

  .

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Check the PIR Sensor values on serial monitor

Arduino Code:

```
#define PIR 33
pinMode(PIR,INPUT);
int pir_status = digitalRead(PIR);
  if (pir_status == HIGH)
      {
    Serial.println("Motion detected!");
    loopDelay=3000;}
  else {
    Serial.println("No Motion detected!");
    loopDelay = 1000;
          }
    delay(loopDelay);
```

Observations:

Result:

8. Gesture Sensor Module (APDS9960)

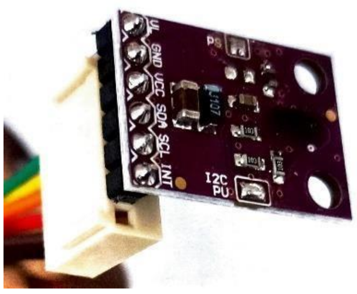
Aim: To interface ESP32 Microcontroller with Gesture Sensor (APDS9960**)** for Gesture Detection.

Components Required:

- GISMO V1 Board and Gesture Sensor (APDS9960)
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

Gesture Sensor



GISMO VI Connection

Sensor signal	ESP32 pin
INT	GPIO23
SCL	GPIO22
SDA	GPIO21
VCC	3.3V
GND	GND

- **Working principle:**

The Gesture Sensor (APDS-9960) is a multipurpose sensor that can be used for Ambient Light, RGB Sensing, Proximity Sensing, and Gesture Detection. This sensor tests the gesture sensing abilities of the APDS-9960. This sensor interfaces with I2C communication protocol and waits for gesture events to calculate the direction of the swipe (up, down, left, right) and displays it on a serial monitor. To perform a near gesture, hold your hand over the sensor and move it close to the sensor (within 2 inches). Hold your hand there for at least 1 second and move it away. To perform a FAR gesture, hold your hand within 2 inches of the sensor for at least 1 second and then move it above (out of range) of the sensor.

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code for identifying the gesture positions.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Tests the gesture sensing abilities of the APDS-9960 to calculate the direction of the swipe (up, down, left, right) and display the direction on the serial monitor.

Arduino Code:

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

// Pins
#define APDS9960_INT 23 // Needs to be an interrupt pin

// Constants

// Global Variables
SparkFun_APDS9960 apds = SparkFun_APDS9960();
int isr_flag = 0;

void setup() {

  // Set interrupt pin as input
  pinMode(APDS9960_INT, INPUT);

  // Initialize Serial port
  Serial.begin(115200);
  Serial.println();
  Serial.println(F("------------------------------------"));
  Serial.println(F("SparkFun APDS-9960 - GestureTest"));
  Serial.println(F("------------------------------------"));

  // Initialize interrupt service routine
  attachInterrupt(APDS9960_INT, interruptRoutine, FALLING);

  // Initialize APDS-9960 (configure I2C and initial values)
  if ( apds.init() ) {
    Serial.println(F("APDS-9960 initialization complete"));
  } else {
    Serial.println(F("Something went wrong during APDS-9960 init!"));
  }

  // Start running the APDS-9960 gesture sensor engine
  if ( apds.enableGestureSensor(true) ) {
  Serial.println(F("Gesture sensor is now running"));
  } else {
    Serial.println(F("Something went wrong during gesture sensor init!"));
  }
  if (apds.setGestureGain(GGAIN_2X))
    {
      Serial.println("Gesture Gain Set");
    }
   else
  {
    Serial.println(F("Something went wrong during gesture sensor init!"));
  }
}

void loop() {
  if( isr_flag == 1 ) {
    detachInterrupt(0);
    handleGesture();
    isr_flag = 0;
    attachInterrupt(APDS9960_INT, interruptRoutine, FALLING);
  }
}

void interruptRoutine() {
  isr_flag = 1;
}

void handleGesture() {
    if ( apds.isGestureAvailable() ) {
    switch ( apds.readGesture() ) {
    case DIR_UP:
        Serial.println("UP");
        break;
      case DIR_DOWN:
        Serial.println("DOWN");
        break;
      case DIR_LEFT:
        Serial.println("LEFT");
        break;
      case DIR_RIGHT:
        Serial.println("RIGHT");
        break;
      case DIR_NEAR:
        Serial.println("NEAR");
        break;
      case DIR_FAR:
        Serial.println("FAR");
        break;
      default:
        Serial.println("NONE");
    }
  }
}
```

Observations:

Result:

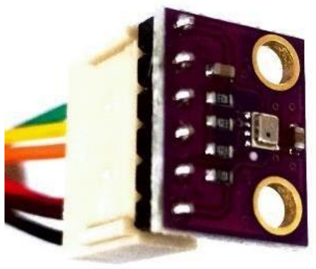
9 Environment Monitoring Module(BMP280)

Aim: To interface ESP32 Microcontroller with BMP280 sensor for measuring the temperature, atmospheric pressure, and altitude.

Components Required:
- GISMO V1 Board and BMP280 senor
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

BMP 280 sensor



GISMO VI connections

Sensor signal	ESP32 pin
SDO	NC
CSB	NC
SDA	GPIO21
SCL	GPIO22
GND	GND
VCC	3.3V

Working Principle:

The BMP280 is an absolute barometric pressure sensor. Its small size and low power consumption makes it to use in battery-powered devices such as mobile phones, GPS modules or watches. The BMP280 is based on Bosch's proven piezo-resistive pressure sensor technology featuring high accuracy and linearity as well as long-term stability and high EMC robustness. Numerous device operation options guarantee for highest flexibility. This sensor can measure barometric pressure and temperature with very good accuracy. Because pressure changes with altitude we can also use it as an altimeter with ±1 meter accuracy. Accuracy for barometric pressure is ±1 hPa and ±1.0°C for temperature. This sensor interface with microcontroller is via I2C or SPI communication protocols.

Procedure:
1. Open Arduino IDE
2. Create a new file and write the required code for measuring the temperature, atmospheric pressure and altitude.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Measure temperature, atmospheric pressure and altitude using BMP280 sensor and display it on a serial monitor.

Arduino Code:

```
#include <Wire.h>
#include <Adafruit_BMP280.h>
```

```
Adafruit_BMP280 bmp; // I2C

#define SEALEVELPRESSURE_HPA (1013.25)
float tempC,tempF,atPressure,altitude,humidity;
void setup() {
  Serial.begin(115200);
  Serial.println(F("BMP280 test"));

  if (!bmp.begin(BMP280_ADDRESS_ALT, BMP280_CHIPID)) {
      Serial.println(F("Could not find a valid BMP280 sensor, check wiring or "
                    "try a different address!"));
    while (1) delay(10);
  }

  /* Default settings from datasheet. */
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode.
*/
                  Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
                  Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
                  Adafruit_BMP280::FILTER_X16,      /* Filtering. */
                  Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */
}

void loop() {
    printValues();
    delay(2000);
}
void printValues(){

  Serial.print("Temperature = ");
  tempC = bmp.readTemperature();
//Serial.print(bme.readTemperature());
Serial.print(tempC);
Serial.println(" deg C");


Serial.print("Temperature = ");
Serial.print(1.8*tempC+32);
Serial.println(" deg F");

Serial.print("Pressure = ");
atPressure=bmp.readPressure()/100.0F;
Serial.print(atPressure);
Serial.println(" hPa");


Serial.print("Altitude = ");
altitude=bmp.readAltitude(SEALEVELPRESSURE_HPA);
Serial.print(altitude);
Serial.println(" m");
```

```
    Serial.println();

    }
```

Observations:

Result:

10. Heart Rate Monitoring Module (MAX30102)

Aim: To interface ESP32 Microcontroller with heartbeat (beats per minute) using MAX30102 sensor.

Components Required:

- GISMO VI Board with MAX30102 Sensor
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

MAX30102 Heart Monitoring sensor

GISMO VI connections

Sensor signal	ESP32 pin
VIN	3.3V
SCL	GPIO22
SDA	GPIO21
INT	NC
IRD	NC
RD	NC
GND	GND

Working principle :

The MAX30102 works by pulse detection through light is called Photoplethysmogram. It is used for Heart Rate Measurement and Pulse Oximetry (measuring the oxygen level of the blood. For the Heart rate measurement, the oxygenated haemoglobin (HbO2) in the arterial blood has taken. This blood has the characteristic of absorbing IR light. The redder the blood (the higher the haemoglobin, the more IR light is absorbed. As the blood is pumped through the finger with each heartbeat, the amount of reflected light changes, creating a changing waveform at the output of the photodetector. If IR light falls on the photodetector, the sensor measures the Heartbeat. Pulse oximetry is based on the principle that the amount of RED and IR light absorbed varies depending on the amount of oxygen in your blood. The deoxygenated blood absorbs more RED light (660nm), while oxygenated blood absorbs more IR light (880nm). By measuring the ratio of IR and RED light received by the photodetector, the oxygen level (SpO2) in the blood is calculated.

Features:

- Heart rate monitor and pulse oximeter sensor in an LED reflective solution
- Tiny 5.6mm x 3.3mm x 1.55mm 14-pin optical module
- Integrated cover glass for optimal, robust performance
- Ultra-low power operation for mobile devices
- Programmable sample rate and LED current for power saving

- Low-power heart rate monitor (<1mW)
- Ultra-low shutdown current (0.7µA Typ.)
- Fast data output capability
- High sample rates Robust motion artifact resilience
- High SNR
- -40°C to +85°C operating temperature range

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code.
3. Add the Spark Fun MAX3010x Pulse and Proximity Sensor Library by SparkFun Electronics libraries to the file.
4. Connect GISMO VI board to laptop, compile and upload code to the board.
5. Measure the BPM values on serial monitor.

Arduino Code:

```
/*
   Optical Heart Rate Detection (PBA Algorithm) using the MAX30105 Breakout
   By: Nathan Seidle @ SparkFun Electronics
   Date: October 2nd, 2016
   https://github.com/sparkfun/MAX30105_Breakout

   This is a demo to show the reading of heart rate or beats per minute (BPM) using
   a Penpheral Beat Amplitude (PBA) algorithm.

   It is best to attach the sensor to your finger using a rubber band or other tightening
  device. Humans are generally bad at applying constant pressure to a thing. When
  you
   press your finger against the sensor it varies enough to cause the blood in your
   finger to flow differently which causes the sensor readings to go wonky.

   Hardware Connections (Breakoutboard to Arduino):
   -5V = 5V (3.3V is allowed)
   -GND = GND
   -SDA = A4 (or SDA)
   -SCL = A5 (or SCL)
   -INT = Not connected

   The MAX30105 Breakout can handle 5V or 3.3V I2C logic. We recommend
  powering the board with 5V
   but it will also run at 3.3V.
  */

  #include <Wire.h>
  #include "MAX30105.h"

  #include "heartRate.h"
```

```
MAX30105 particleSensor;

const byte RATE_SIZE = 8; //Increase this for more averaging. 4 is good.
byte rates[RATE_SIZE]; //Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred

float beatsPerMinute;
int beatAvg;

void setup()
{
  Serial.begin(115200);
  Serial.println("Initializing...");

  // Initialize sensor
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port,
400kHz speed
  {
    Serial.println("MAX30105 was not found. Please check wiring/power. ");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with steady pressure.");

  particleSensor.setup(); //Configure sensor with default settings
particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate
sensor is running
  particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
}

void loop()
{
  long irValue = particleSensor.getIR();

  if (checkForBeat(irValue) == true)
  {
    //We sensed a beat!
    long delta = millis() - lastBeat;
    lastBeat = millis();

    beatsPerMinute = 60 / (delta / 1000.0);

    if (beatsPerMinute < 255 && beatsPerMinute > 20)
    {
      rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in the array
      rateSpot %= RATE_SIZE; //Wrap variable

      //Take average of readings
      beatAvg = 0;
      for (byte x = 0 ; x < RATE_SIZE ; x++)
```

```
        beatAvg += rates[x];
      beatAvg /= RATE_SIZE;
    }
  }

 // Serial.print("IR=");
  //Serial.print(irValue);
  //Serial.print(", BPM=");
  //Serial.print(beatsPerMinute);
  Serial.print("Avg BPM=");
  Serial.print(beatAvg);

  if (irValue < 50000)
    Serial.print(" No finger?");

  Serial.println();
}
```

Observations

Result:

11. Multi-Axis Accelerometer Module MPU6050

Aim: To interface ESP32 Microcontroller with MPU6050 Sensor to detect fall forward, backward, left, or right using multi-axes accelerometer MPU6050.

Components Required:

- GISMO VI Board and MPU6050 Sensor
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

Accelerometer MPU 6050

GISMO VI connections

MPU6050 pin	ESP32 pin
VCC	3.3V
GND	GND
SCL	GPIO22
SDA	GPIO21

Specifications

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

Working principle:

The MPU6050 combines both gyroscope and an accelerometer, using which we can measure rotation along all three axes, static acceleration due to gravity, as well as motion, shock, or dynamic acceleration due to vibration. MEMS (Micro Electro Mechanical Systems) accelerometer consists of a micro-machined structure built on top of a silicon wafer. This structure is suspended by polysilicon springs. It allows the structure to deflect at the time when the acceleration is applied on the axis. Due to deflection the capacitance between fixed plates and plates attached to the suspended structure is changed. This change in capacitance is proportional to the acceleration on that axis. The sensor processes this change in capacitance and converts it into an analog output voltage. While accelerometers measure linear acceleration, MEMS gyroscopes measure angular rotation. It is an inexpensive 6-axis Motion racking chip that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor (DMP) all in a small 4mm x 4mm package. It can measure angular momentum or rotation along all the three axes, the static acceleration due to gravity, as well as dynamic acceleration resulting from motion, shock, or vibration. The module comes with an on-board LD3985 3.3V regulator. The MPU6050 consumes less than 3.6mA during measurements and only 5μA during idle. This low power consumption allows the implementation in battery driven devices. In addition, the module has a power LED that lights up when the module is powered.

Micro-electromechanical systems (MEMS) is a process technology used to create tiny integrated.

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Detect fall forward, backward, left or right using multi-axes accelerometer MPU6050 on serial monitor.

Arduino Code:

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
MPU6050 mpu;
float baseline[3];
float features[3];
float motion_threshold = 0.7;

void setup() {

   Wire.begin();
   Serial.begin(115200);
   mpu.initialize();
   calibrate();

  }

void loop() {

   float ax,ay,az;
   mpu_read(&ax,&ay,&az);
   ax=ax-baseline[0];
   ay=ay-baseline[1];
   az=az-baseline[2];

   mpu_record();

 delay(2000);

  }

void mpu_read(float *ax,float *ay,float *az) {
   int16_t _ax, _ay, _az, _gx, _gy, _gz;
```

```
    mpu.getMotion6(&_ax, &_ay, &_az, &_gx, &_gy, &_gz);
    *ax = _ax/16384.0;
    *ay = _ay/16384.0;
    *az = _az/16384.0;

}

void calibrate(){
  float ax,ay,az;
  for(int i=0;i < 10;i++){
    mpu_read(&ax,&ay,&az);
    delay(100);
      }
  baseline[0]=ax;
  baseline[1]=ay;
  baseline[2]=az;
}


void mpu_record(){
  float ax,ay,az;
  float aax,aay,aaz;
  String position;
    mpu_read(&ax,&ay,&az);
    ax = ax - baseline[0];
    ay = ay - baseline[1];
    az = az - baseline[2];

   features[0] = ax;
   features[1] = ay;
   features[2] = az;

  Serial.print(features[0]);
  Serial.print(" ");
  Serial.print(features[1]);
  Serial.print(" ");
  Serial.println(features[2]);

  aax=fabs(ax);
  aay=fabs(ay);
  aaz=fabs(az);

  Serial.print(aax);
  Serial.print(" ");
  Serial.print(aay);
  Serial.print(" ");
  Serial.println(aaz);

  position="Upright";
  if(aax > motion_threshold)
  {
    if(ax > 0)
    position="Left";
    else
    position="Right";
  }
```

```
if(aay > motion_threshold)
{
  if(ay > 0)
  position="Backward";
  else
  position="Forward";
}



Serial.println(position);

}
```

Observations:

Result:

12. Magnetic Switch Interface

Aim: Controlling the Magnetic Switch to open or close door based on switch status with ESP-32 Microcontroller on GISMO VI Board.

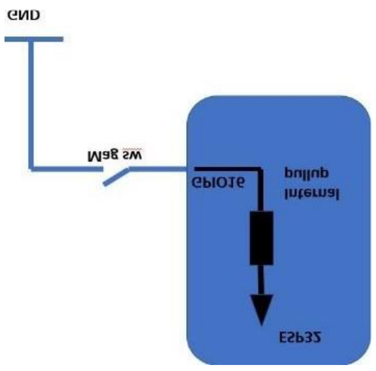
Components Required:

- GISMO VI Board
- Laptop with Arduino IDE
- USB cable
- Magnetic Switch

On-board pin assignments:

Magnetic switch **GISMO VI connections**

Magnetic Switch sensor signal	ESP32 pin
Sensor o/p contact 1	GPIO16
Sensor o/p contact 2	GND



Working principle:

- If you want to Apply Extra Security for your house then this Magnetic Switch is needed for you. This magnetic switch is used for triggering an alarm or ON/OFF light inside a cupboard sliding door
- MC-38 Wired Door Window Sensor Magnetic Switch Home Alarm System can be used as a door or window security system.
- It produces the signal when moved away from each other which can be fed to the microcontroller to perform the desired action as per requirement.
- This sensor is suitable to use for triggering an alarm or ON/OFF light inside a cupboard sliding door. This wired sensor is triggered by the magnet. When the

magnet is closed by, the circuit is closed or open if the magnet is far from the sensor.

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code for controlling the Magnetic Switch.
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Check the door opening and closing based on switch status on GISMO VI at GPIO Pin 16 of ESP-32 Microcontroller on serial monitor.

Arduino Code:

```
#define magSW 16
void setup( )
{
        // put your setup code here, to run once:
pinMode(magSW,INPUT_PULLUP);
Serial.begin(115200);
}
void loop( )
 {
  // put your main code here, to run repeatedly:
  int sw_status;
  sw_status = digitalRead(magSW);
  if (sw_status)
  Serial.println("Window open");
  else
  Serial.println("Window closed");
  delay(1000);
  }
```

Observations:

Result:

13. OLED Display interface

Aim: To interface ESP32 Microcontroller with OLED to read the text.

Components Required:

- GISMO VI Board and OLED
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

OLED

GISMO VI connections

Sensor signal	ESP32 pin
SDA	GPIO21
SCL	GPIO22
VCC	3.3V
GND	GND

Working principle:

OLED displays are electronic visual panels that harness organic light-emitting diodes (which, of course, is what the acronym OLED stands for) for their core illumination power. OLED is a type of electroluminescent display technology, in which an organic material layer generates light when molecules in the diode are agitated by an electric current.

As with many other display types, an OLED array can be used to present images, text, video and more on a screen or panel of almost any size, and the technology has been especially prevalent on the high-end home entertainment market over the past few years. Thanks to the unique strengths they deliver in terms of power and performance, OLED screens are also in increasingly widespread use as performance display tools across all industries and sectors today.

To control the OLED display you need the adafruit_SSD1306.h and the Adafruit_GFX.h libraries. Follow the next instructions to install those libraries.

1. Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.
2. Type "**SSD1306**" in the search box and install the SSD1306 library from Adafruit.
3. After installing the SSD1306 library from Adafruit, type "**GFX**" in the search box and install the library.

Methods used in OLED for displaying the text:

- display.clearDisplay() – all pixels are off
- display.drawPixel(x,y, color) – plot a pixel in the x,y coordinates
- display.setTextSize(n) – set the font size, supports sizes from 1 to 8
- display.setCursor(x,y) – set the coordinates to start writing text
- display.print(“message”) – print the characters at location x,y
- display.display() – call this method for the changes to make effect

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code for displaying the text on OLED
3. Connect GISMO VI board to laptop, compile and upload code to the board.
4. Check the concatenated string of Hello with count on OLED display.

Arduino code:

```
#include <Wire.h>
#include "SSD1306.h"

SSD1306 display(0x3c,21,22);
String myString;
int count;
  void setup() {
    OLEDInit();
  }

  void loop() {
  OLEDUpdate();
  count++;
  if(count==10) count=0;
  delay(1000);
}

void OLEDInit(){
  display.init();
  display.setFont(ArialMT_Plain_24);
}

void OLEDUpdate(){
  display.clear();
  myString = "Hello " + String(count);
  display.drawString(0,0,myString);
  display.display();
}
```

Observations:

Result:

14. IR Sensor Interface

Aim: To interface ESP32 Microcontroller with IR Remote based Control to have Local control of an appliance.

Components Required:

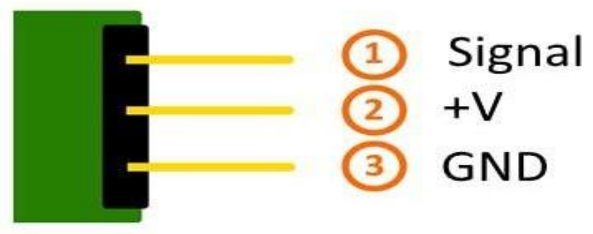
- GISMO VI Board and IR Sensor
- Laptop with Arduino IDE
- USB cable

On-board pin assignments:

IR receiver

GISMO VI connections

Sensor signal	ESP32 pin
OUT	GPIO4
GND	GND
VCC	3.3V



Working principle:

An infrared sensor is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, which can be detected by an infrared sensor. The emitter is simply an IR LED and the detector is simply an IR photodiode that is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received.
There are different types of infrared transmitters depending on their wavelengths, output power and response time. An IR sensor consists of an IR LED and an IR Photodiode, together they are called as Photocoupler or Optocoupler. Infrared receivers or infrared sensors detect the radiation from an IR transmitter. IR receivers come in the form of photodiodes and phototransistors. Infrared Photodiodes are different from normal photo diodes as they detect only infrared radiation IR LED is one kind of transmitter that emits

IR radiations. This LED looks similar to a standard LED and the radiation which is generated by this is not visible to the human eye. Infrared receivers mainly detect the radiation using an infrared transmitter. These infrared receivers are available in photodiodes form. IR Photodiodes are dissimilar as compared with usual photodiodes because they detect simply IR radiation. Different kinds of infrared receivers mainly exist depending on the voltage, wavelength, package, etc. Once it is used as the combination of an IR transmitter & receiver, then the receiver's wavelength must equal the transmitter. Here, the transmitter is IR LED whereas the receiver is IR photodiode. The infrared photodiode is responsive to the infrared light that is generated through an infrared LED. The resistance of photo-diode & the change in output voltage is in proportion to the infrared light obtained.

Procedure:

1. Open Arduino IDE
2. Create a new file and write the required code.
3. Connect GISMO VI board to laptop, compile, and upload code to the board.
4. Detect fall forward, backward, left, or right using multi-axes accelerometer MPU6050 on serial monitor.

Arduino Code:

```
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>

// An IR detector/demodulator is connected to GPIO pin 16(D5 on a NodeMCU
// board).
// Note: GPIO 16 won't work on the ESP8266 as it does not have interrupts.
const uint16_t kRecvPin = 4;

IRrecv irrecv(kRecvPin);

decode_results results;

void setup() {
  pinMode(2,OUTPUT);
  pinMode(13,OUTPUT);
  Serial.begin(115200);
  irrecv.enableIRIn();  // Start the receiver
  while (!Serial)  // Wait for the serial connection to be establised.
    delay(50);
  Serial.println();
  Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
  Serial.println(kRecvPin);
}

void loop() {
  if (irrecv.decode(&results)) {
    // print() & println() can't handle printing long longs. (uint64_t)
    long btnValue = results.value;
    serialPrintUint64(btnValue);
```

```
     Serial.println("");
     switch(btnValue){
       case 16718055:
       case 16736925:
       case 16712445:
       digitalWrite(2,HIGH);
       digitalWrite(13,HIGH);
       Serial.println("Up arrow");
       break;
       case 16730805:
       case 16754775:
       case 16720605:
       digitalWrite(2,LOW);
       digitalWrite(13,LOW);
       Serial.println("Down arrow");
       break;
       case 16738455:
       case 16728765:
       case 16753245:
       Serial.println("Local Control");
       break;
       case 16756815:
       case 16732845:
       case 16769565:
       Serial.println("Cloud Control");
       break;
       default:
       Serial.println("Not a valid key");
       break;
     }
     irrecv.resume();  // Receive the next value

   }
   delay(100);
 }
```

Observations:

Result:

TASK 2: Mobile App Development

1 Develop apps with simple UI

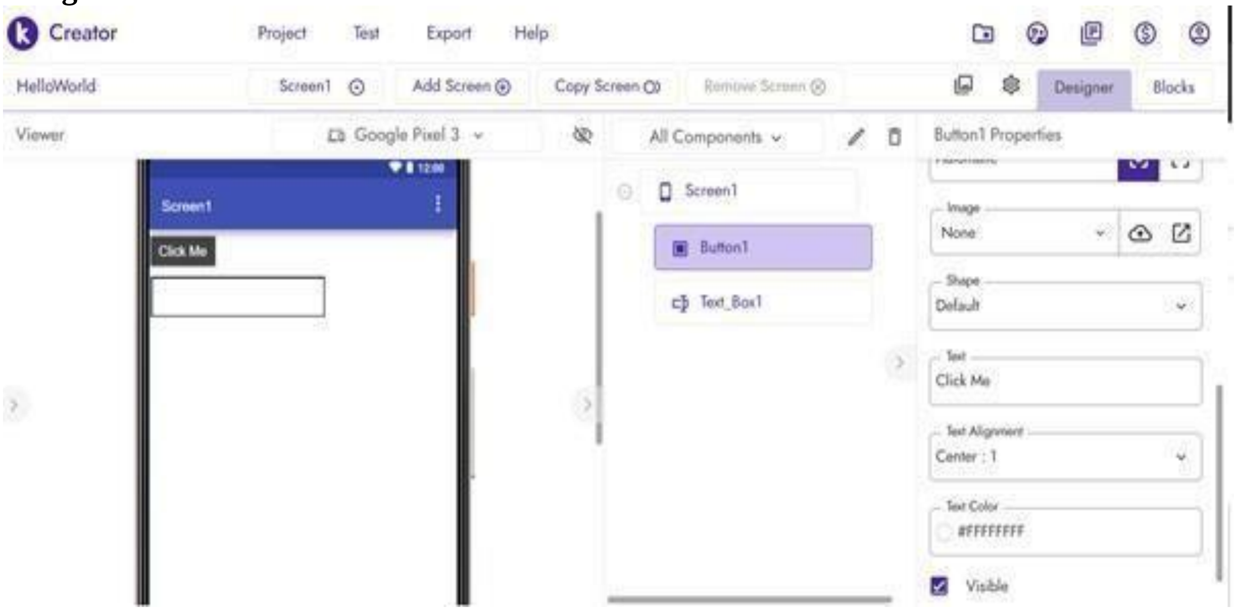
Aim: To develop an app with simple User Interface using Kodular Creator Platform.

Equipment Required:

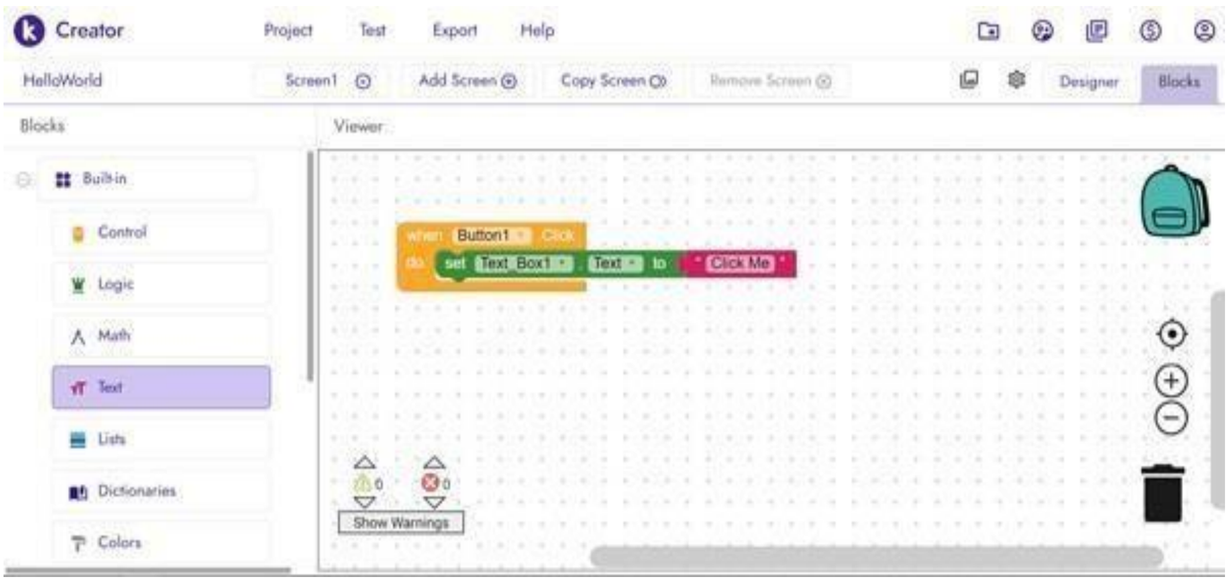
1. Laptop with Wi-Fi connection
2. Mobile Phone with Kodular Companion app.

Design Diagrams:

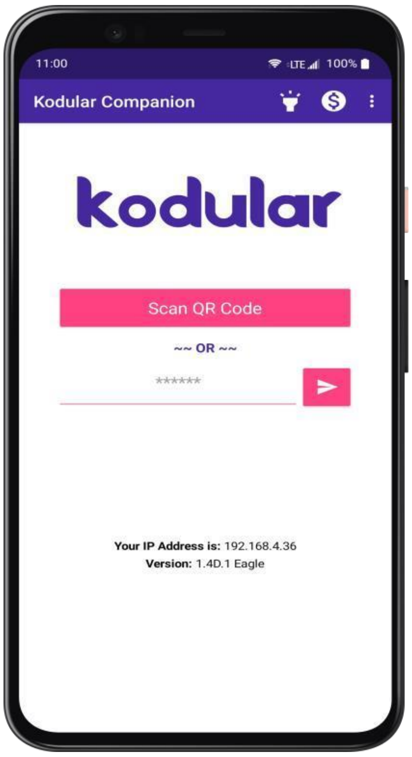
Designer View:



Block View:



Testing and Development:



Procedure:

1. Open Kodular creator in web browser and create a project
2. In designer view, find the Button component from the Palette under the User Interface category and drag it into the Viewer. Next find the Text Box component and drag it into the Viewer
3. Click on the Button1 component. The Button1 properties open up on the right side. Go to Button1 Text property and change it to Click Me
4. Go to blocks, Click on the Button1 component. The Button1 functions open up on the Viewer
5. Drag and drop the When Button1 Click block onto the Viewer
6. Click on the Text_Box1 component. Drag and drop the Set Text_Box1. Text block onto the Viewer
7. Go to the Text Blocks group. Drag and drop an empty Text and write "Click Me" in the block
8. Download the Kodular Companion app from google play store and open it.
9. In Kodular Creator, go to *"Test"* palette and choose *"Connect to companion"* from drop down menu. Scan the QR code using companion app. (Both laptop and mobile phone should be in the same Wi-Fi network)

Observations:

Result:

2 Mobile app to push pull data from the cloud database

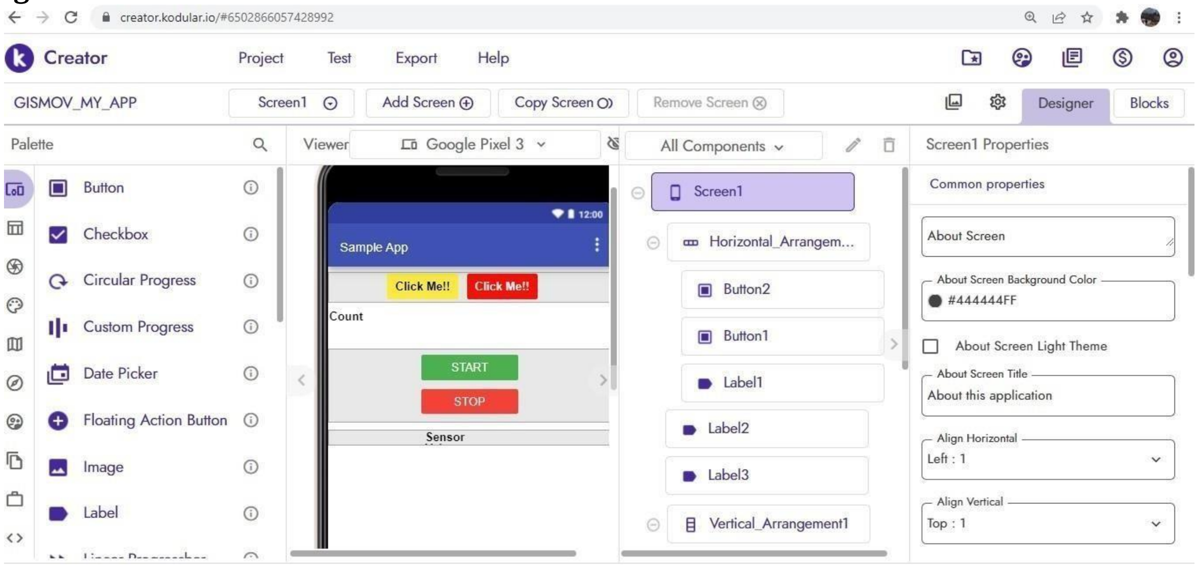
Aim: To develop a mobile app which can push and pull data from the firebase

Equipment Required:

1. Laptop with Wi-Fi connection and google firebase account
2. Mobile phone with Kodular Companion app.
3. GISMO VI board and USB cable.

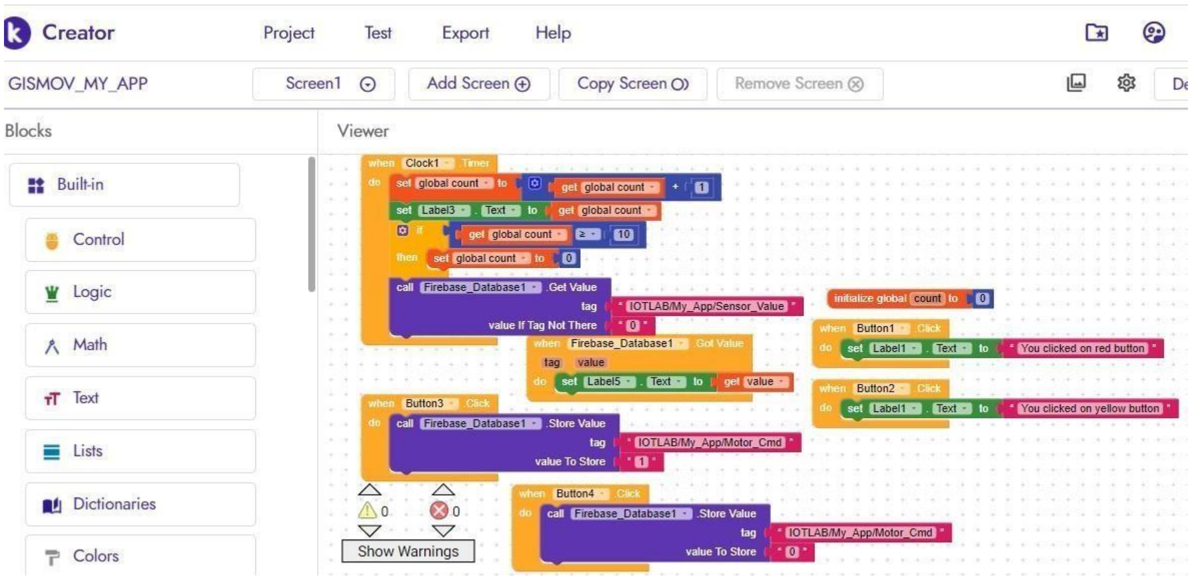
Mobile App:

Designer View:



About Screen
About Screen Background Color: #444444FF
About Screen Light Theme: ☐
About Screen Title: About this application
Align Horizontal: Left : 1
Align Vertical: Top : 1

Block View:



```
when Clock1 .Timer
do  set global count to ( get global count + 1 )
    set Label3 . Text to get global count
    if get global count ≥ 10
    then set global count to 0
    call Firebase_Database1 .Get Value
                        tag "IOTLAB/My_App/Sensor_Value"
          value If Tag Not There "0"

initialize global count to 0

when Firebase_Database1 .Got Value
  tag  value
do  set Label5 . Text to get value

when Button1 .Click
do  set Label1 . Text to "You clicked on red button"

when Button2 .Click
do  set Label1 . Text to "You clicked on yellow button"

when Button3 .Click
do  call Firebase_Database1 .Store Value
                        tag "IOTLAB/My_App/Motor_Cmd"
              value To Store "1"

when Button4 .Click
do  call Firebase_Database1 .Store Value
                        tag "IOTLAB/My_App/Motor_Cmd"
              value To Store "0"
```

Arduino Code:

```
#include <WiFi.h>
#include "FirebaseESP32.h"
#define FIREBASE_HOST "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Do not includehttps://
in FIREBASE_HOST
#define FIREBASE_AUTH "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
#define WIFI_SSID "xxxxxxxxxxxxxxxxxx"
#define WIFI_PASSWORD "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
//Define Firebase Data object
FirebaseData firebaseData;
#define ledPin 2
float value = 0.0;
void setup() {                  // put your setup code here, to run once:
pinMode(ledPin,OUTPUT);
Serial.begin(115200);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
}
void loop() {                     // put your main code here, to run repeatedly:
String ledCmdFull;
String ledCmd;
int finalCmd;
Firebase.getString(firebaseData,"IOTLAB/My_App/Motor_Cmd",ledCmdFull);
ledCmd = ledCmdFull.substring(2,ledCmdFull.length()-2);
finalCmd = ledCmd.toInt();
Serial.println(finalCmd);
if(finalCmd == 1)
{
  digitalWrite(ledPin,HIGH);
  Serial.println("LED ON");
}
else
{
  digitalWrite(ledPin,LOW);
  Serial.println("LED OFF");
}
 value = value +1.54;
 if(value > 10.0) value = 0.0;
   Firebase.setFloat(firebaseData,"IOTLAB/My_App/Sensor_Value",value);
   delay(1000);
}
```

Procedure:

1. Open Arduino IDE and write the code. Give Wi-Fi and firebase credentials in the code.
2. Connect GISMO VI board to laptop, compile and upload code to the board.
3. Check the data in firebase.
4. Open Kodular creator and develop app (give firebase credentials in the app).
5. In Kodular Creator, go to *"Test"* palette and choose *"Connect to companion"* from drop down menu. Scan the QR code using companion app. (Both laptop and mobile phone should be in the same Wi-Fi network)

Observations:

Results:

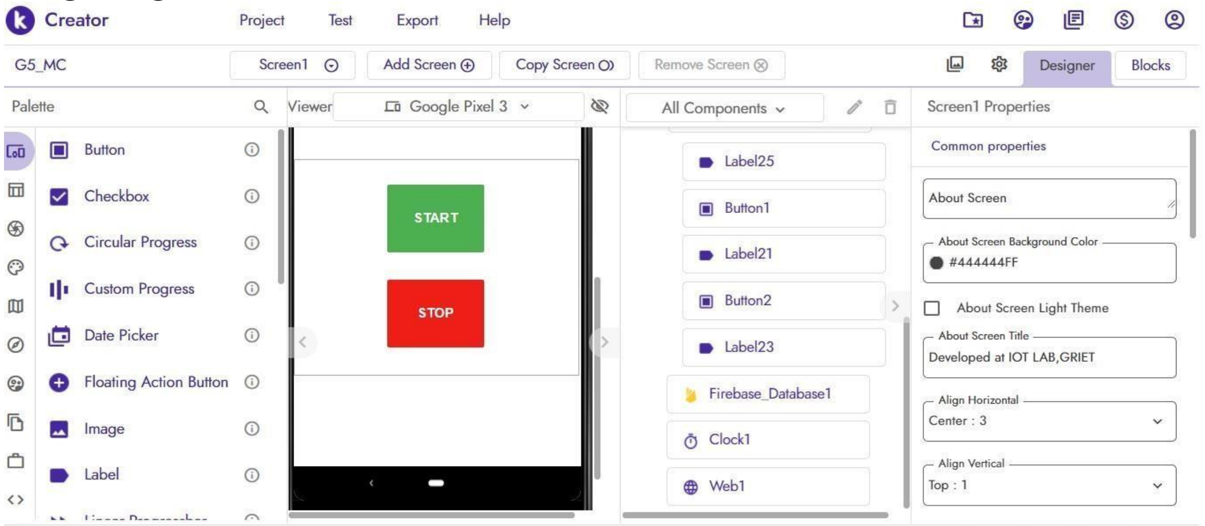
3 Mobile app to Push actuator commands to the cloud database

Aim: To design a mobile app to push actuator commands to the firebase.

Equipment Required:

1. Laptop with Wi-Fi connection
2. Mobile phone with Kodular companion app

Design Diagrams:



About Screen

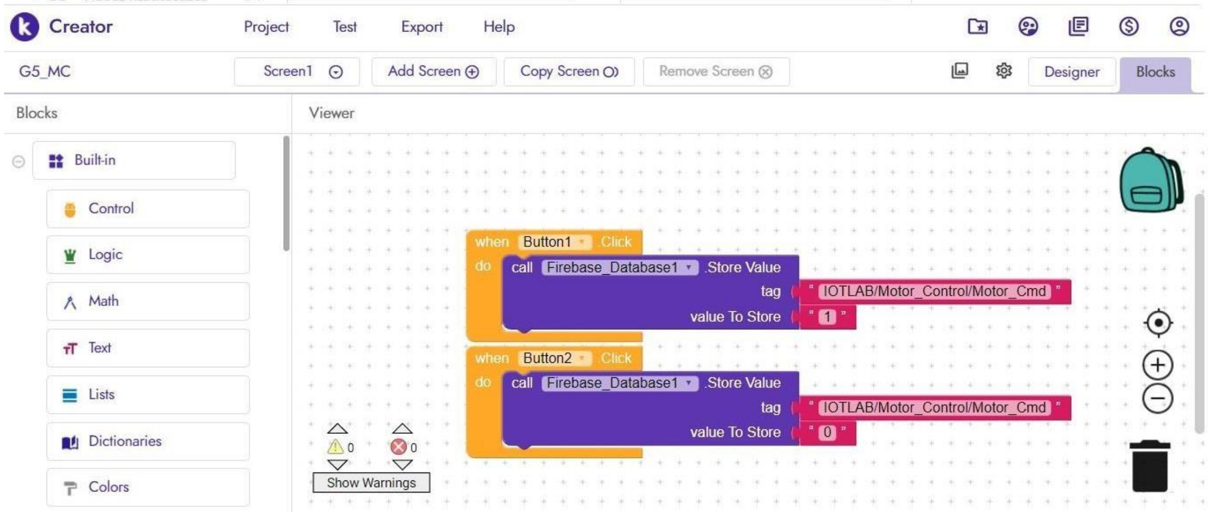
About Screen Background Color: #444444FF

About Screen Light Theme

About Screen Title: Developed at IOT LAB,GRIET

Align Horizontal: Center : 3

Align Vertical: Top : 1



```
when Button1 .Click
do  call Firebase_Database1 .Store Value
                        tag  " IOTLAB/Motor_Control/Motor_Cmd "
              value To Store  " 1 "

when Button2 .Click
do  call Firebase_Database1 .Store Value
                        tag  " IOTLAB/Motor_Control/Motor_Cmd "
              value To Store  " 0 "
```

Procedure:

1. Open Kodular creator in web browser and create a project.
2. Design an app with firebase component to push “Start” and “Stop” data.
3. Open firebase and check the data.

Observation:

Result:

TASK 3: IOT Projects

1 Home Security System with IoT interface

Aim: To detect the presence of intruders by checking the opening/closing of windows using PIR, magnetic switch sensors and update the alarm status in Firebase and mobile app

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with PIR Sensor, Magnetic Switch and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO VI board to laptop, compile and upload the code to the board.
4. Check temperature, humidity values on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to detect the presence of intruders by checking the opening/closing of windows using PIR, magnetic switch sensors

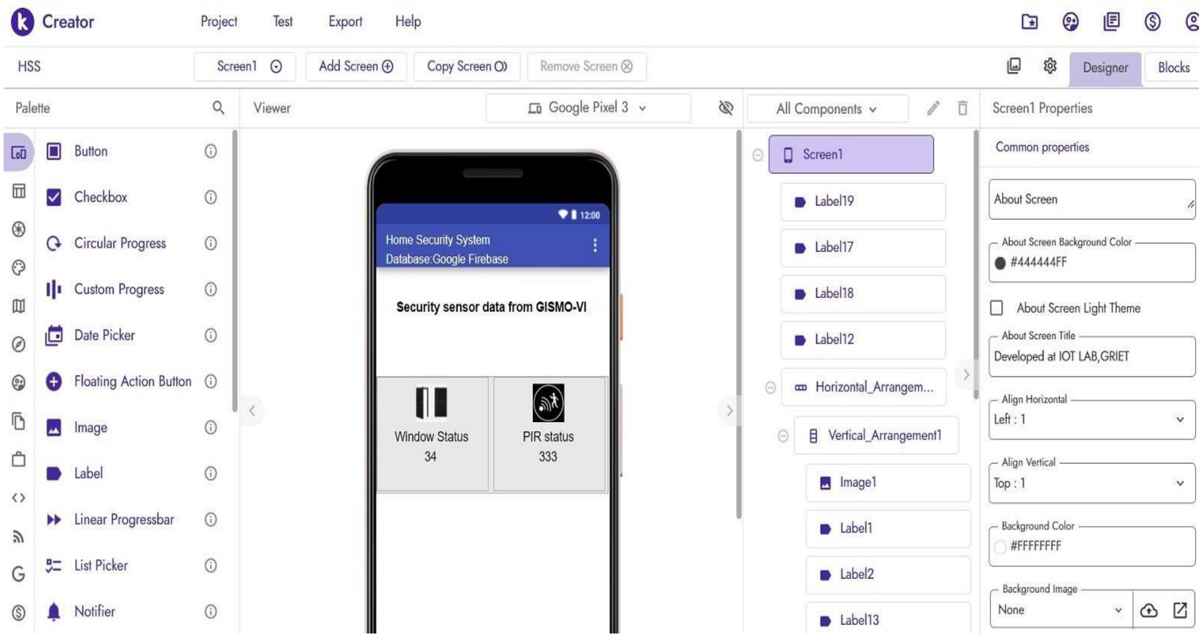
Arduino Code:

```
#define magSwitch 16
#define PIR 33
#define LED 2
#include "Credentials.h"
String magswStatus;
String PIRStatus;
int loopDelay = 1000;
void OLEDInit();
void OLEDUpdate();
void WiFiinit();
void FirebaseInit();
void FirebaseWrite();
String tag1 = "IOTLAB/HSS/MagswStatus";
String tag2 = "IOTLAB/HSS/PIRSwitch";
void setup() {
  // put your setup code here, to run once:
  pinMode(magSwitch,INPUT_PULLUP);
  pinMode(PIR,INPUT);
  pinMode(LED,OUTPUT);
  Serial.begin(115200);
  OLEDInit();
  WiFiInit();
  FirebaseInit();
  // Initial warmup period of 1 minute for PIR sensor
  for(int i = 0; i < 60; i++){
  digitalWrite(2,!digitalRead(2));
    delay(1000);
  }
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
// Mag switch reading
 int sw_status = digitalRead(16);
 if(sw_status == HIGH){
 Serial.println("OPEN");
 magswStatus = "Window_Open";}
 else{
 Serial.println("CLOSED");
 magswStatus = "Window_Closed";}
// PIR sensor reading
  int pir_status = digitalRead(PIR);
  if (pir_status == HIGH) {
    Serial.println("Motion detected!");
    PIRStatus = "Motion_detected";
    loopDelay = 3000;
    }
  else {
    Serial.println("No Motion");
    PIRStatus = "No_Motion";
    loopDelay = 1000;
    }
OLEDUpdate();
FirebaseWrite();
delay(loopDelay);
}
void OLEDInit(){
  display.init();
  display.setFont(ArialMT_Plain_16);
  display.clear();
  display.drawString(0,0,"Warming up...");
  display.display();
}
void OLEDUpdate(){
  display.clear();
  display.drawString(0,0,magswStatus);
  display.drawString(0,30,PIRStatus);
  display.display();
}
void WiFiInit(){
  pinMode(2,OUTPUT);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    digitalWrite(2,!digitalRead(2));
    delay(300);
  }


  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
```

```
    Serial.println();
     }
  void FirebaseInit(){
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    Firebase.reconnectWiFi(true);
  }
  void FirebaseWrite(){
    Firebase.setString(firebaseData,tag1,magswStatus);
    Firebase.setString(firebaseData,tag2,PIRStatus);
   }
```

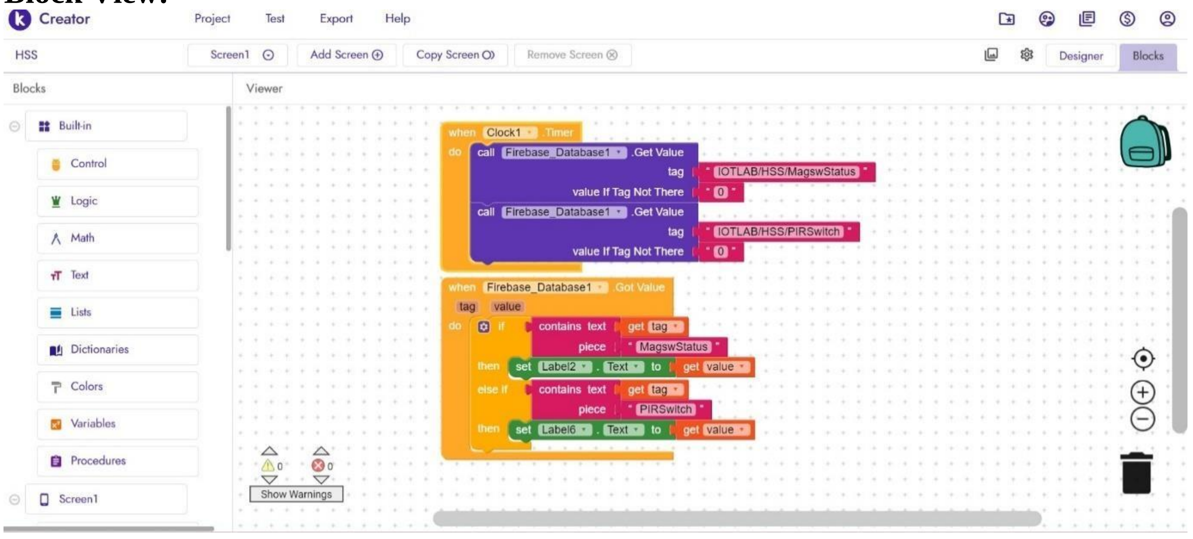
Mobile App:
Designer View:



Home Security System
Database:Google Firebase
Security sensor data from GISMO-VI
Window Status
34
PIR status
333

About Screen
About Screen Background Color: #444444FF
About Screen Light Theme
About Screen Title: Developed at IOT LAB,GRIET
Align Horizontal: Left : 1
Align Vertical: Top : 1
Background Color: #FFFFFFFF
Background Image: None

Block View:



```
when Clock1 .Timer
do  call Firebase_Database1 .Get Value
                              tag  "IOTLAB/HSS/MagswStatus"
          value If Tag Not There  "0"
    call Firebase_Database1 .Get Value
                              tag  "IOTLAB/HSS/PIRSwitch"
          value If Tag Not There  "0"

when Firebase_Database1 .Got Value
  tag  value
do  if       contains text  get tag
                    piece  "MagswStatus"
    then  set Label2 . Text to  get value
    else if  contains text  get tag
                    piece  "PIRSwitch"
    then  set Label6 . Text to  get value
```

Observations:

Result:

2. Smart Garden with IOT Interface

Aim: To Detect the level of soil moisture and switch on motor based on limit checking, then accordingly transfer soil moisture data and motor status to Firebase and display in mobile app.

Equipment Required:
1. Laptop with Wi-Fi connection
2. GISMO VI board with soil moisture sensor and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:
1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO V I board with soil moisture sensor and relay to laptop, compile andupload the code to the board.
4. Check soil moisture value and motor status on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to display soil moisture and motor status

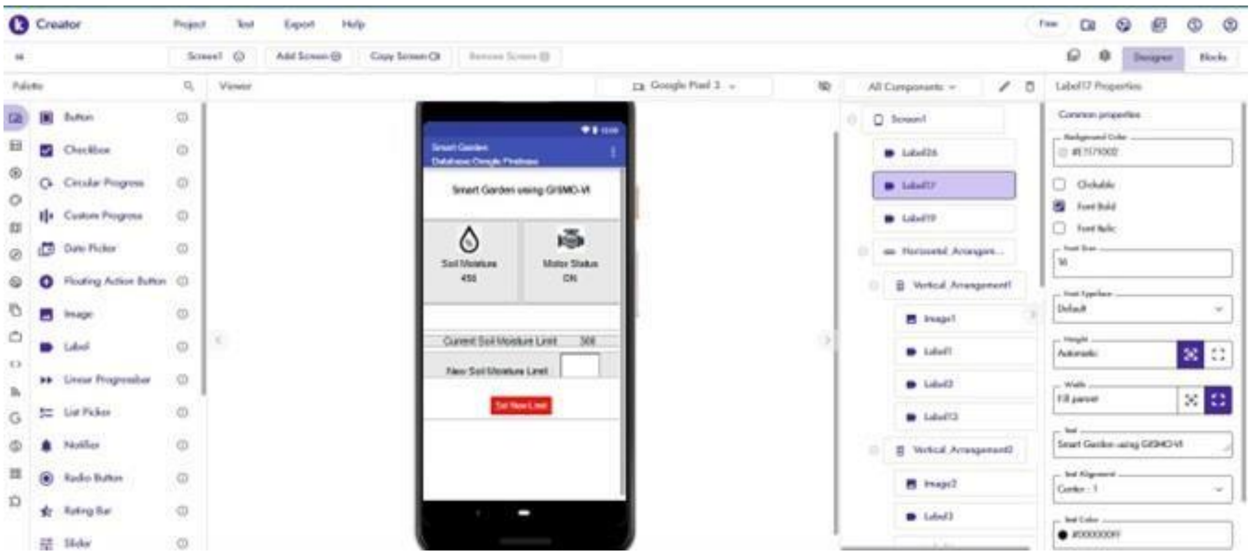
Arduino Code:

```
/* Soil Moisture sensor pin: GPIO34
* Relay pin: GPIO13
#include "FirebaseESP32.h"
#define FIREBASE_HOST "xxxxxxxxxxxxxxxxxxxxxxxxxxx" //Do not include https:// in
FIREBASE_HOST
#define FIREBASE_AUTH "xxxxxxxxxxxxxxxxxxxxxxxxx"
#define WIFI_SSID "xxxxxxxxxxxxxxx"
#define WIFI_PASSWORD "xxxxxxxxxxxxxxxxxxxxxx"
//Define Firebase Data object
FirebaseData firebaseData;

#include <Wire.h>
#include "SSD1306.h"

SSD1306  display(0x3c, 21, 22);

#define smPin 34
#define relayPin 13int
smValue;
int smLimit = 3000;
int delayTime = 2000;
String smValueCloud;
void setup() {
// put your setup code here, to run once:
```

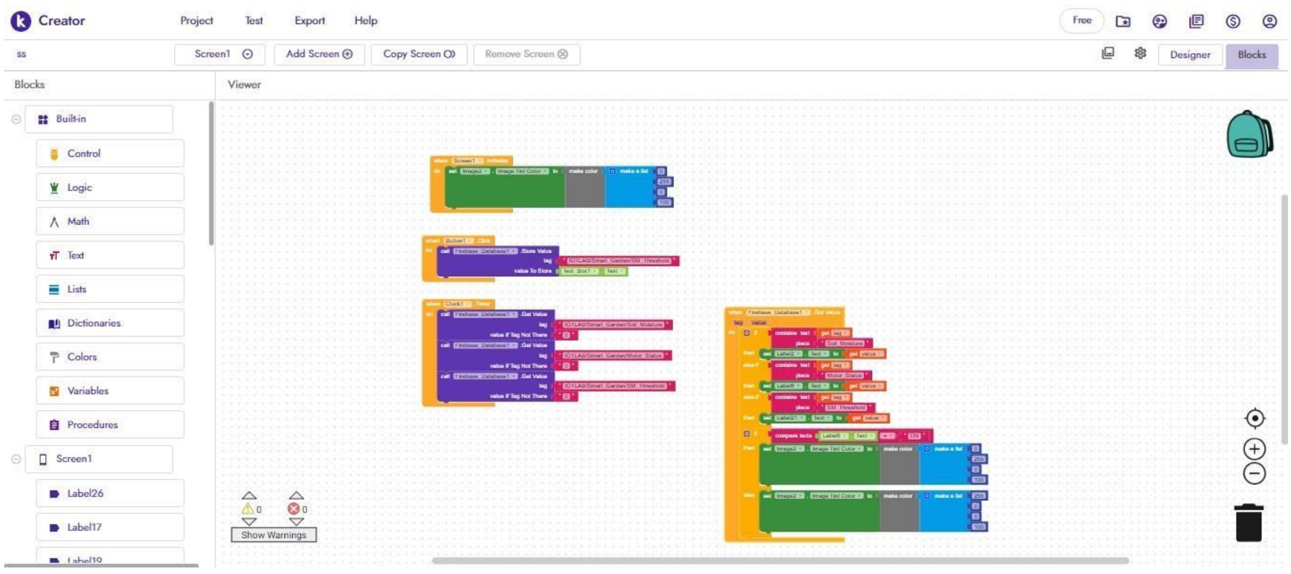
```
pinMode(relayPin,OUTPUT);
Serial.begin(115200);

//Connecting to Wi-Fi network
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED)
{
Serial.print(".");
delay(300);
}
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
Firebase.reconnectWiFi(true);

display.init();
display.setFont(ArialMT_Plain_16);

}

void loop() {
// put your main code here, to run repeatedly:
String smCloudFull;
String smCloud;
if(Firebase.getString(firebaseData,"IOTLAB/Smart_Garden/SM_Threshold",smCloudFull)){
smCloud = smCloudFull.substring(2,smCloudFull.length()-2);
smLimit = smCloud.toInt();
Serial.println(smLimit);
}
smValue = 4095 - analogRead(smPin);
Serial.print("Soil Moisture = ");
Serial.println(smValue);
Firebase.setInt(firebaseData,"IOTLAB/Smart_Garden/Soil_Moisture",smValue);

display.clear();
String myString = "SM = ";
myString.concat(smValue);
display.drawString(0,0,myString);

if (smValue < smLimit){
digitalWrite(relayPin,HIGH);
Serial.println("Motor turned ON");
Firebase.setString(firebaseData,"IOTLAB/Smart_Garden/Motor_Status","ON");
display.drawString(0,30,"MOTOR ON ");
}
else{
digitalWrite(relayPin,LOW);
Serial.println("Motor turned
OFF");Firebase.setString(firebaseData,"IOTLAB/Smart_Garden/Motor_Status","OFF");
display.drawString(0,30,"MOTOR OFF");
}
display.display();
delay(delayTime);
}
```

Mobile App:

Designer View:



Block View:



Observations:

Result:

3. IR Remote based motor Control with IoT Interface

Aim: To design a mobile app to control a motor, transfer the motor status from mobile app to firebase and GISMO board

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with relay and OLED display.
3. Android mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

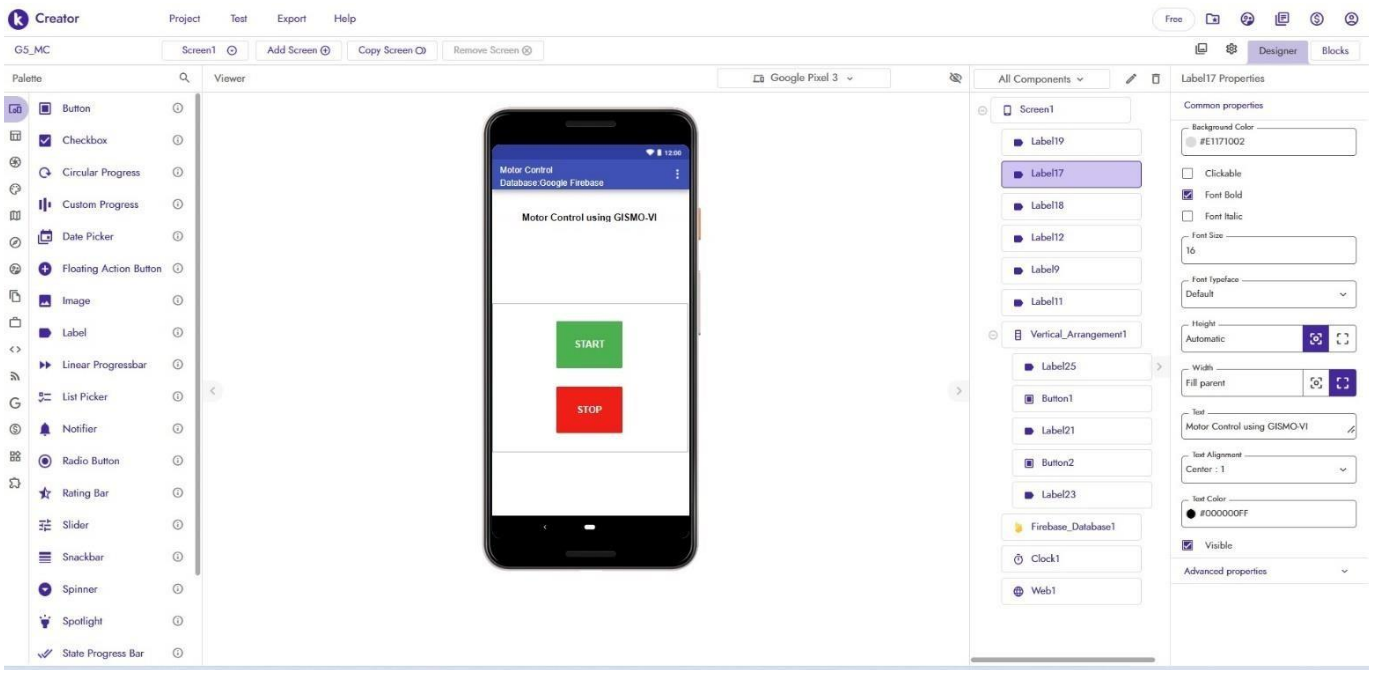
1. Open Arduino IDE, create a new file and write the code.
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO VI board with onboard relay to laptop, compile and upload the code to the board.
4. Check the status on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to display motor status in Google Firebase database

Arduino Code:

```
#include <Wire.h>
#include "SSD1306.h"
SSD1306  display(0x3c, 21, 22);

#define motorPin 13

#include "FirebaseESP32.h"
#define FIREBASE_HOST "xxxxxxxxxxxxxxxxxxxxxxx" //Do not include https:// in FIREBASE_HOST
#define FIREBASE_AUTH "xxxxxxxxxxxxxxxxxxxxxxx"
#define WIFI_SSID "xxxxxxxxxxxxxxxxx"
#define WIFI_PASSWORD "xxxxxxxxxxxxxxxxxxx"
//Define Firebase Data object
FirebaseData firebaseData;

int delayTime = 2000;

void setup() {
  // put your setup code here, to run once:
  pinMode(motorPin,OUTPUT);
  Serial.begin(115200);

 //Connecting to Wi-Fi network
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
  display.init();
 display.setFont(ArialMT_Plain_16);

}

void loop() {
  // put your main code here, to run repeatedly:

String motorCmdFull;
String motorCmd;
int finalCmd;
display.clear();
Firebase.getString(firebaseData,"IOTLAB/Motor_Control/Motor_Cmd",motorCmdFull);
motorCmd = motorCmdFull.substring(2,motorCmdFull.length()-2);
finalCmd = motorCmd.toInt();
if(finalCmd == 1)
{
  digitalWrite(motorPin,HIGH);
  Serial.println("Motor ON");
  display.drawString(20,0,"MOTOR ON");
}
else
{
  digitalWrite(motorPin,LOW);
  Serial.println("Motor OFF");
  display.drawString(20,0,"MOTOR OFF");

}
display.display();
delay(delayTime);
}
```

Mobile App:

Designer View:



Label17 Properties

Common properties

Background Color: #E1171002

Clickable: ☐

Font Bold: ✓

Font Italic: ☐

Font Size: 16

Font Typeface: Default

Height: Automatic

Width: Fill parent

Text: Motor Control using GISMO-VI

Text Alignment: Center : 1

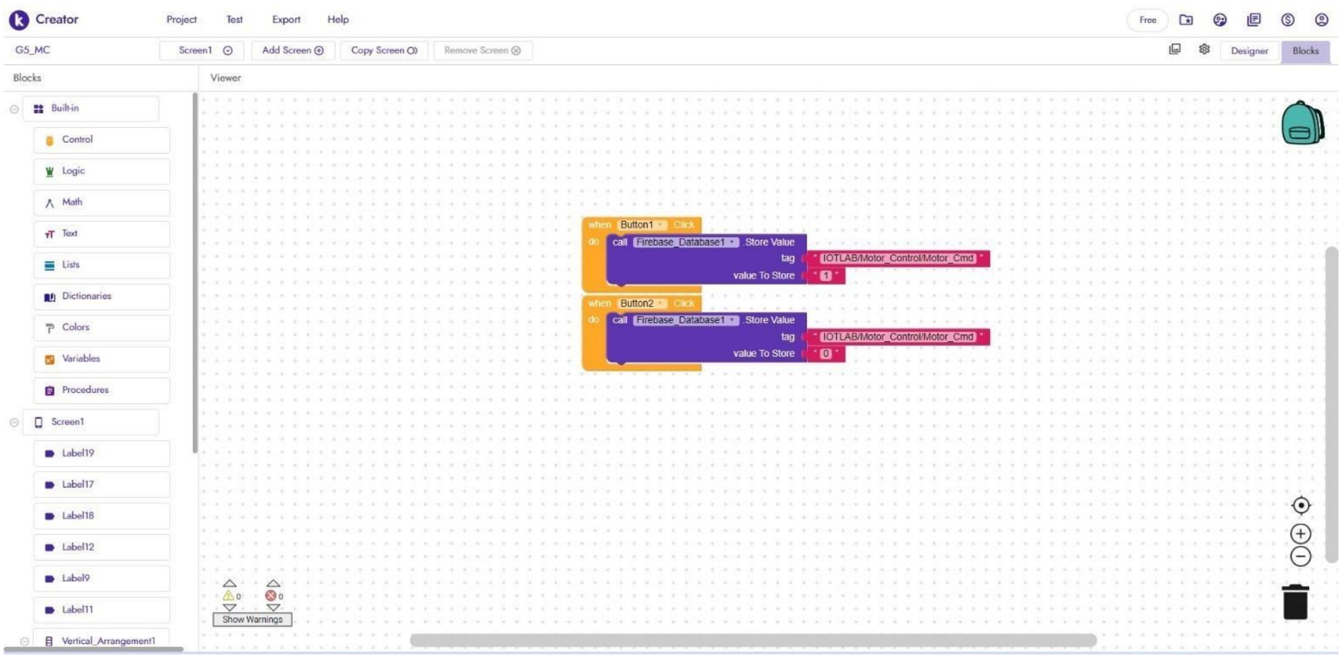
Text Color: #000000FF

Visible: ✓

Advanced properties

Components: Screen1, Label19, Label17, Label18, Label12, Label9, Label11, Vertical_Arrangement1, Label25, Button1, Label21, Button2, Label23, Firebase_Database1, Clock1, Web1

Block View:



```
when Button1 .Click
do  call Firebase_Database1 .Store Value
                          tag  " IOTLAB/Motor_Control/Motor_Cmd "
                value To Store  " 1 "
when Button2 .Click
do  call Firebase_Database1 .Store Value
                          tag  " IOTLAB/Motor_Control/Motor_Cmd "
                value To Store  " 0 "
```

Observations:

Result:

4 IOT based Remote Range Meter

Aim: To detect range at which object is present using Ultrasonic sensor HCSR04 and then accordingly transfer range data to Firebase and display in mobile app.

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with HCSR04 ultrasonic sensor and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

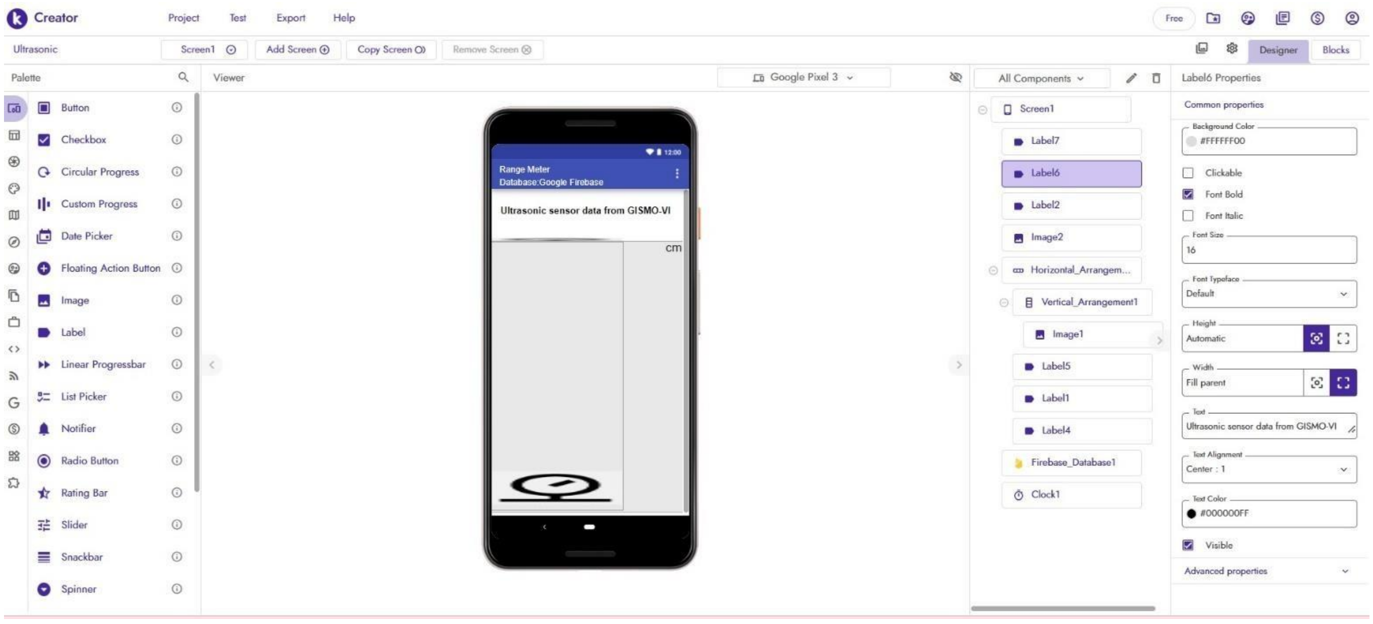
1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO VI board with HCSR04 ultrasonic sensor module to laptop, compile and upload the code to the board.
4. Check range on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to display the range at which the object is present.

Arduino Code:

```
#include "FirebaseESP32.h"
#define FIREBASE_HOST "xxxxxxxxxxxxxxxxxxxxxxxxx" //Do not include https:// in
FIREBASE_HOST
#define FIREBASE_AUTH "xxxxxxxxxxxxxxxxxxxxxxxxx"
#define WIFI_SSID "xxxxxxxxxxxxxxxxxx"
#define WIFI_PASSWORD "xxxxxxxxxxxxx"
//Define Firebase Data object
FirebaseData firebaseData;
#include <Wire.h>
#include "SSD1306.h"
SSD1306  display(0x3c, 21, 22);
// defines pins numbers
const int trigPin = 25;
const int echoPin = 26;
// defines variables
long duration;
int distance;
int delayTime = 2000;
void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(115200); // Starts the serial communication
//Connecting to Wi-Fi network
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED)
{
Serial.print(".");
delay(300);
}
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
Firebase.reconnectWiFi(true);
display.init();
display.setFont(ArialMT_Plain_24);
}
void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance= duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);
Firebase.setInt(firebaseData,"IOTLAB/Range_Meter/Range",distance);
display.clear(); String
myString = "";
myString.concat(distance);
myString.concat(" cm");
  display.drawString(0,0,myString);
display.display();
delay(delayTime);
}
```

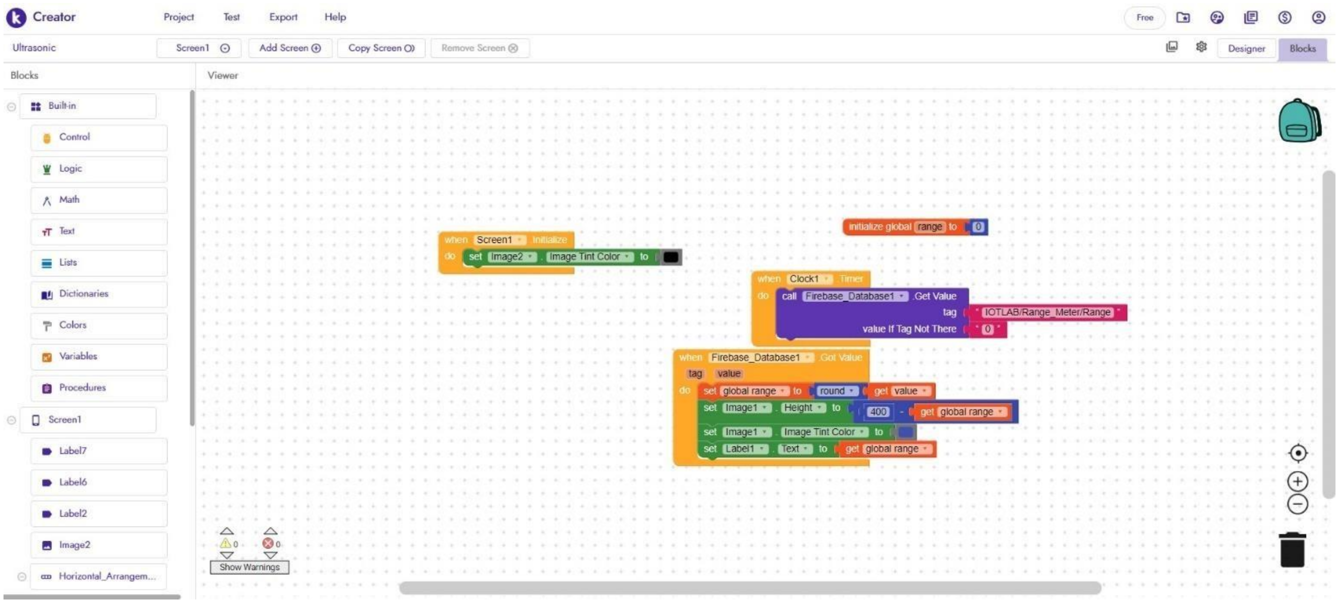
Mobile App:

Designer View:



Background Color: #FFFFFF00
Clickable: ☐
Font Bold: ☑
Font Italic: ☐
Font Size: 16
Font Typeface: Default
Height: Automatic
Width: Fill parent
Text: Ultrasonic sensor data from GISMO-VI
Text Alignment: Center : 1
Text Color: ● #000000FF
Visible: ☑

Block View:



```
initialize global range to 0

when Screen1.Initialize
do  set Image2.Image Tint Color to ■

when Clock1.Timer
do  call Firebase_Database1.Get Value
                             tag  "IOTLAB/Range_Meter/Range"
             value If Tag Not There  "0"

when Firebase_Database1.Got Value
  tag  value
do  set global range to round  get value
    set Image1.Height to  400 - get global range
    set Image1.Image Tint Color to ■
    set Label1.Text to  get global range
```

Observations:

Result:

5. Fall Detection using IoT

Aim: To detect fall forward, backward, left or right using multi-axes accelerometer MPU6050 and transfer fall status to Firebase and display in mobile app

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with HCSR04 ultrasonic sensor and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

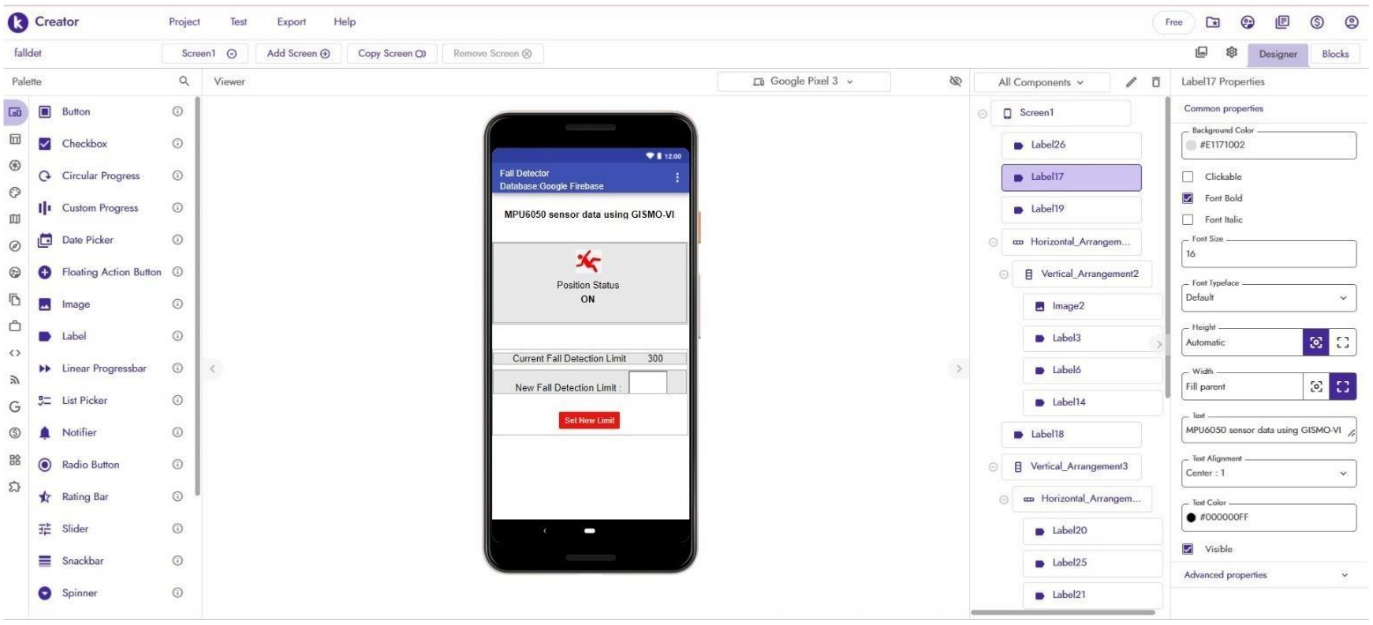
1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO V I board with MPU6050 multi-axes accelerometer sensor module to laptop, compile and upload the code to the board.
4. Check range on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to detect fall forward, backward, left or right using multi-axes accelerometer MPU6050.

Arduino Code:

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
MPU6050 mpu;
#include "Credentials.h"
float baseline[3];
float features[3];
float motion_threshold = 0.7;
void OLEDInit();
void WiFiInit();
void FirebaseInit();
void FirebaseRead();
String position;
void setup() {
   Wire.begin();
   Serial.begin(115200);
   mpu.initialize();
   calibrate();
   OLEDInit();
   WiFiInit();
   FirebaseInit();
       }

void loop() {
     float ax,ay,az;
   //FirebaseRead();
   mpu_read(&ax,&ay,&az);
   ax=ax-baseline[0];
   ay=ay-baseline[1];
```

```
    az=az-baseline[2];
       mpu_record();
    delay(2000);
       }

void mpu_read(float *ax,float *ay,float *az) {
    int16_t _ax, _ay, _az, _gx, _gy, _gz;
    mpu.getMotion6(&_ax, &_ay, &_az, &_gx, &_gy, &_gz);
    *ax = _ax/16384.0;
    *ay = _ay/16384.0;
    *az = _az/16384.0;

}

void calibrate(){
  float ax,ay,az;
  for(int i=0;i < 10;i++){
    mpu_read(&ax,&ay,&az);
    delay(100);
      }
  baseline[0]=ax;
  baseline[1]=ay;
  baseline[2]=az;
}


void mpu_record(){
  float ax,ay,az;
  float aax,aay,aaz;
  String position;
    mpu_read(&ax,&ay,&az);
    ax = ax - baseline[0];
    ay = ay - baseline[1];
   az = az - baseline[2];
   features[0] = ax;
   features[1] = ay;
   features[2] = az;

  Serial.print(features[0]);
  Serial.print(" ");
  Serial.print(features[1]);
  Serial.print(" ");
  Serial.println(features[2]);
  aax=fabs(ax);
  aay=fabs(ay);
  aaz=fabs(az);
  Serial.print(aax);
  Serial.print(" ");
  Serial.print(aay);
  Serial.print(" ");
  Serial.println(aaz);
  position="Upright";
  if(aax > motion_threshold)
  {
    if(ax > 0)
    position="Left";
    else
    position="Right";
  }
  if(aay > motion_threshold)
  {
    if(ay > 0)
    position="Backward";
    else
    position="Forward";
  }

     Serial.println(position);
  Firebase.setString(firebaseData,"IOTLAB/Fall_Detector/Position",position);
  display.clear();
  display.drawString(30,0,"Position:");
  display.drawString(30,30,position);
  display.display();

  }
  void OLEDInit(){
  display.init();
  display.setFont(ArialMT_Plain_16);
}
void WiFiInit(){
  pinMode(2,OUTPUT);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    digitalWrite(2,!digitalRead(2));
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

}
void FirebaseInit(){
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);

}
void FirebaseRead(){
String fdCloudFull;
String fdCloud;
if(Firebase.getString(firebaseData,"IOTLAB/Fall_Detector/Limit",fdCloudFull)){
fdCloud = fdCloudFull.substring(2,fdCloudFull.length()-2);
motion_threshold = fdCloud.toFloat();
Serial.println(motion_threshold);
}
}
```

Mobile App:
Designer View:



Fall Detector
Database:Google Firebase

MPU6050 sensor data using GISMO-VI

Position Status
ON

Current Fall Detection Limit 300

New Fall Detection Limit :

Set New Limit

Label17 Properties

Common properties

Background Color: #E1171002

Clickable: ☐
Font Bold: ✓
Font Italic: ☐

Font Size: 16

Font Typeface: Default

Height: Automatic

Width: Fill parent

Text: MPU6050 sensor data using GISMO-VI

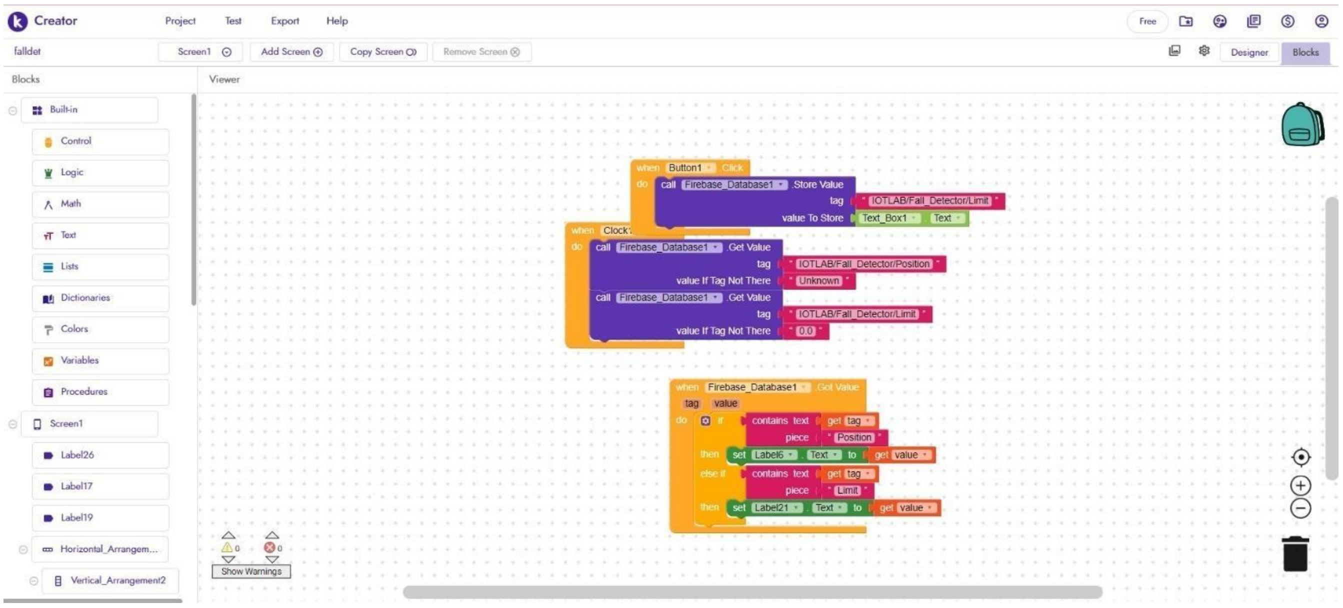
Text Alignment: Center : 1

Text Color: ● #000000FF

Visible: ✓

Advanced properties

Block View:



```
when Button1 .Click
do  call Firebase_Database1 .Store Value
                                tag  " IOTLAB/Fall_Detector/Limit "
                      value To Store  Text_Box1 . Text

when Clock1
do  call Firebase_Database1 .Get Value
                                tag  " IOTLAB/Fall_Detector/Position "
               value If Tag Not There  " Unknown "
    call Firebase_Database1 .Get Value
                                tag  " IOTLAB/Fall_Detector/Limit "
               value If Tag Not There  " 0.0 "

when Firebase_Database1 .Got Value
 tag  value
do  if      contains text  get tag
                   piece  " Position "
    then    set Label6 . Text to  get value
    else if contains text  get tag
                   piece  " Limit "
    then    set Label21 . Text to  get value
```

Observations:

Result:

6. Gesture Recognition using IoT

Aim: To Recognize the gesture positions (left, right, up and down) using APDS9960 sensor and transfer its status to Firebase and display in mobile app.

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with APDS9960 sensor and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO VI board with APDS9960 sensor module to laptop, compile and upload the code to the board.
4. Check range on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to recognize the gesture positions (left, right, up and down)

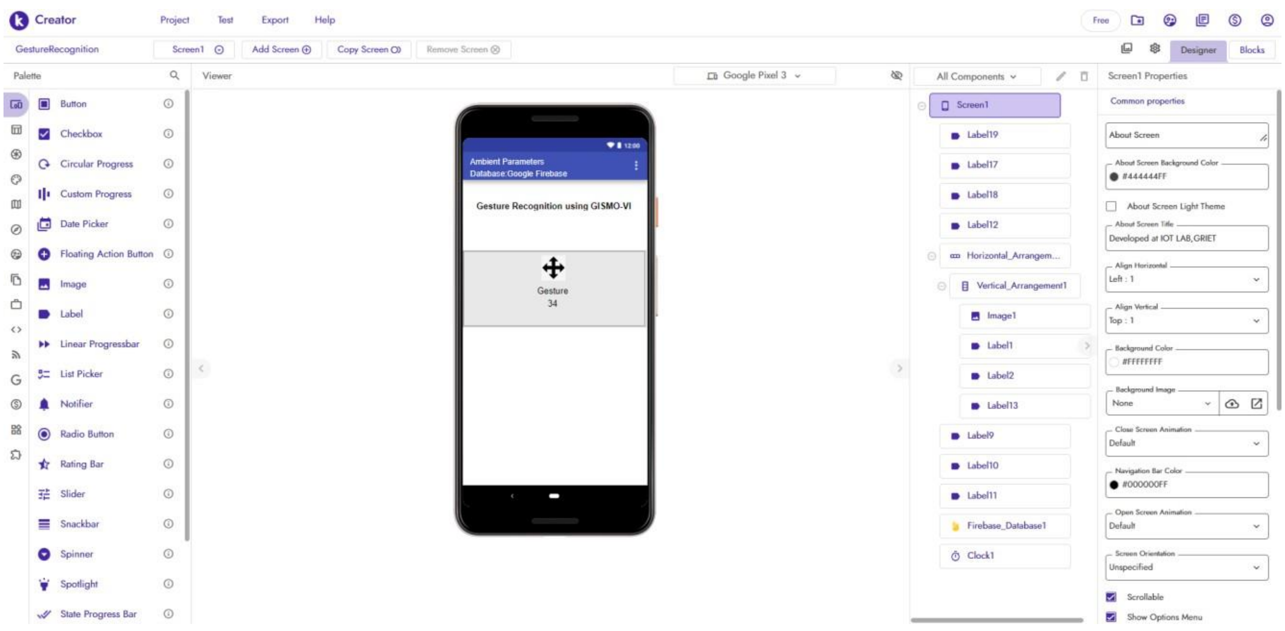
Arduino Code:

```
#include "Credentials.h"
#include <Wire.h>
#include <SparkFun_APDS9960.h>

String tag = "IOTLAB/GestureRecognition/Gesture";

// Pins
#define APDS9960_INT 23 // Needs to be an interrupt pin

// Constants

// Global Variables
SparkFun_APDS9960 apds = SparkFun_APDS9960();
int isr_flag = 0;

void OLEDInit();
void OLEDUpdate();
void WiFiinit();
void FirebaseInit();
void FirebaseWrite();

String gesture = "NONE";

void setup() {

  // Set interrupt pin as input
  pinMode(APDS9960_INT, INPUT);

  // Initialize Serial port
  Serial.begin(115200);
  Serial.println();
  Serial.println(F("------------------------------------"));
  Serial.println(F("SparkFun APDS-9960 - GestureTest"));
  Serial.println(F("------------------------------------"));

  // Initialize interrupt service routine
  attachInterrupt(APDS9960_INT, interruptRoutine, FALLING);

  // Initialize APDS-9960 (configure I2C and initial values)
  if ( apds.init() ) {
    Serial.println(F("APDS-9960 initialization complete"));
  } else {
    Serial.println(F("Something went wrong during APDS-9960 init!"));
  }

  // Start running the APDS-9960 gesture sensor engine
  if ( apds.enableGestureSensor(true) ) {
  Serial.println(F("Gesture sensor is now running"));
  } else {
    Serial.println(F("Something went wrong during gesture sensor init!"));
  }
  if (apds.setGestureGain(GGAIN_2X))
    {
      Serial.println("Gesture Gain Set");
    }
   else
  {
    Serial.println(F("Something went wrong during gesture sensor init!"));
  }

  OLEDInit();
  WiFiInit();
  FirebaseInit();
}

void loop() {
  if( isr_flag == 1 ) {
    detachInterrupt(APDS9960_INT);
    handleGesture();
    OLEDUpdate();
    FirebaseWrite();
    isr_flag = 0;
    attachInterrupt(APDS9960_INT, interruptRoutine, FALLING);
  }
}

void interruptRoutine() {
  isr_flag = 1;
}

void handleGesture() {
    if ( apds.isGestureAvailable() ) {
    switch ( apds.readGesture() ) {
    case DIR_UP:
        Serial.println("UP");
        gesture = "UP";
        break;
      case DIR_DOWN:
        Serial.println("DOWN");
        gesture = "DOWN";
        break;
      case DIR_LEFT:
        Serial.println("LEFT");
        gesture = "LEFT";
        break;
      case DIR_RIGHT:
        Serial.println("RIGHT");
        gesture = "RIGHT";
        break;
      case DIR_NEAR:
        Serial.println("NEAR");
        gesture = "NEAR";
        break;
      case DIR_FAR:
        Serial.println("FAR");
        gesture = "FAR";
        break;
      default:
        Serial.println("NONE");
        gesture = "NONE";
    }
  }
}

void OLEDInit(){
  display.init();
  display.setFont(ArialMT_Plain_24);
}

void OLEDUpdate(){
  display.clear();
  display.drawString(0,0,"Gesture :");
  display.drawString(0,30,gesture);
  display.display();
}
void WiFiInit(){
  pinMode(2,OUTPUT);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    digitalWrite(2,!digitalRead(2));
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
```

```
}
void FirebaseInit(){
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);

}
void FirebaseWrite(){
  Firebase.setString(firebaseData,tag,gesture);

}
```

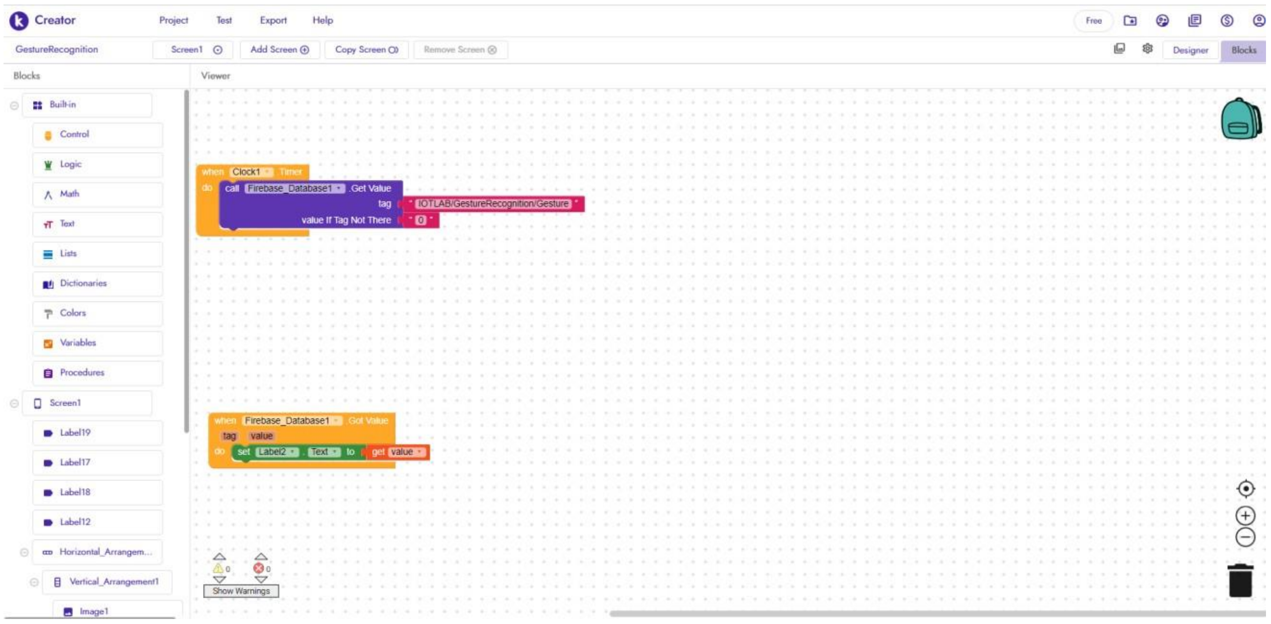
Mobile App:

Designer View:



Screen1 Properties
Common properties
About Screen
About Screen Background Color: #444444FF
☐ About Screen Light Theme
About Screen Title: Developed at IOT LAB,GRIET
Align Horizontal: Left : 1
Align Vertical: Top : 1
Background Color: #FFFFFFFF
Background Image: None
Close Screen Animation: Default
Navigation Bar Color: #000000FF
Open Screen Animation: Default
Screen Orientation: Unspecified
☑ Scrollable
☑ Show Options Menu

Block View:



```
when Clock1 .Timer
do  call Firebase_Database1 .Get Value
                         tag  "IOTLAB/GestureRecognition/Gesture"
        value If Tag Not There  "0"

when Firebase_Database1 .Got Value
  tag  value
do  set Label2 . Text to  get value
```

Observations:

Result:

7. Heart Rate Monitoring using IoT

Aim: To monitor the Heart Rate or heartbeat (beats per minute) using MAX30102 sensor, then accordingly transfer heartbeat data to Firebase and display in mobile app.

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with MAX30102 Sensor and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO VI board to laptop, compile and upload the code to the board.
4. Check BMP values on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to display MAX30102 Sensor (BMP) values

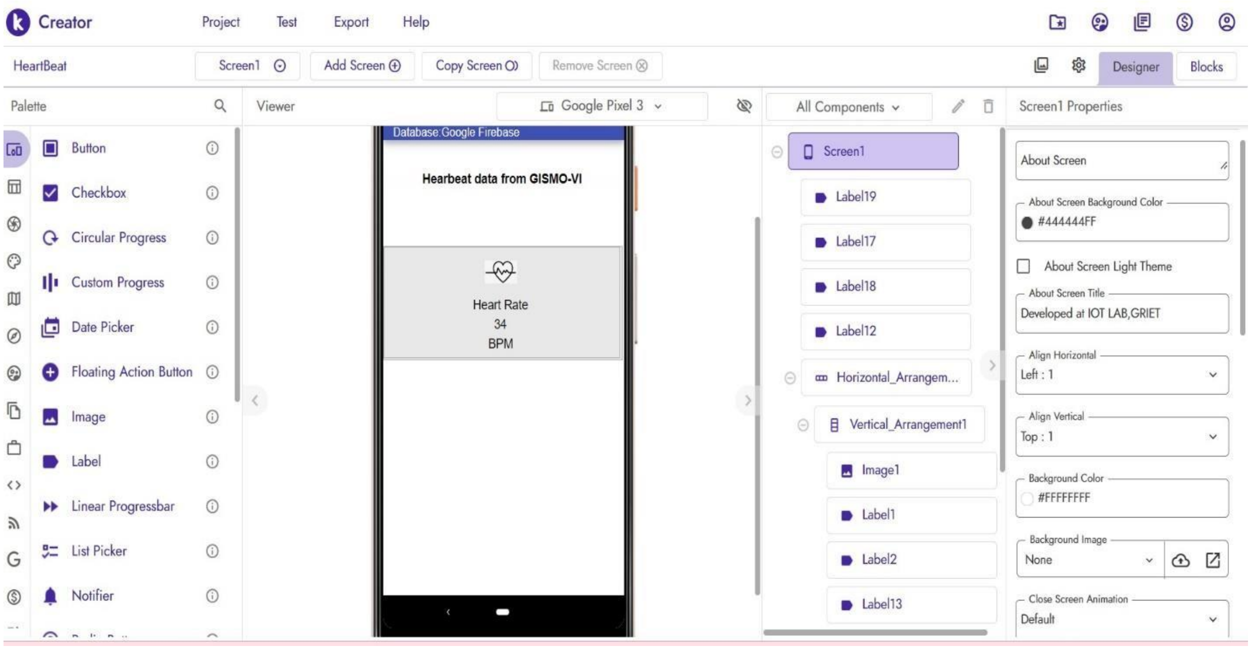
Arduino Code:

```
#include "Credentials.h"
#include "MAX30105.h"
#include "heartRate.h"
MAX30105 particleSensor;
String tag = "IOTLAB/HeartBeat/HR";
const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is good.
byte rates[RATE_SIZE]; //Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred
float beatsPerMinute;
int beatAvg;
int beatCount;
int loopCount;
void WiFiInit();
void OLEDInit();
void OLEDUpdate();
void FirebaseInit();
void FirebaseWrite();
void setup()
{
  Serial.begin(115200);
  Serial.println("Initializing...");
  // Initialize sensor
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port, 400kHz speed
  {
    Serial.println("MAX30105 was not found. Please check wiring/power. ");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with steady pressure.");
  particleSensor.setup(); //Configure sensor with default settings
  particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate sensor is running
  particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
  WiFiInit();
  FirebaseInit();
  OLEDInit();
}
void loop()
{
  long irValue = particleSensor.getIR();
  if (checkForBeat(irValue) == true)
  {
    //We sensed a beat!
    long delta = millis() - lastBeat;
    lastBeat = millis();
    beatsPerMinute = 60 / (delta / 1000.0);

    if (beatsPerMinute < 255 && beatsPerMinute > 20)
    {
      rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in the array
      rateSpot %= RATE_SIZE; //Wrap variable
      //Take average of readings
      beatAvg = 0;
      for (byte x = 0 ; x < RATE_SIZE ; x++)
        beatAvg += rates[x];
      beatAvg /= RATE_SIZE;
           beatCount++;
      if(beatCount == 4)
      {
       Serial.print("Avg BPM=");
       Serial.print(beatAvg);
       OLEDUpdate();
       FirebaseWrite();
       beatCount = 0;
      }
    }
  }
  if (irValue < 50000)
    {
     loopCount++;
     if(loopCount == 10){
      loopCount = 0;
     Serial.println(" No finger?");
     display.clear();
     display.drawString(0,0,"No Finger?");
     display.display();
     FirebaseWriteNF();
    }
    delay(100);
    }
  }
void OLEDInit(){
  display.init();
  display.setFont(ArialMT_Plain_24);
}
void OLEDUpdate(){
  display.clear();
  display.drawString(0,0,"Heart Rate:");
  display.drawString(40,30,String(beatAvg));
```

```
   display.display();
 }
 void WiFiInit(){
   pinMode(2,OUTPUT);
   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
   Serial.print("Connecting to Wi-Fi");
   while (WiFi.status() != WL_CONNECTED)
   {
     Serial.print(".");
     digitalWrite(2,!digitalRead(2));
     delay(300);
   }
   Serial.println();
   Serial.print("Connected with IP: ");
   Serial.println(WiFi.localIP());
   Serial.println();
    }
 void FirebaseInit(){
   Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
   Firebase.reconnectWiFi(true);
 }
 void FirebaseWrite(){
   Firebase.setString(firebaseData,tag,String(beatAvg));
  }
 void FirebaseWriteNF(){
   Firebase.setString(firebaseData,tag,"No_Finger?");
  }
```

Mobile App:

Designer View:



Screen1 Properties

About Screen

About Screen Background Color: #444444FF

About Screen Light Theme

About Screen Title: Developed at IOT LAB,GRIET

Align Horizontal: Left : 1

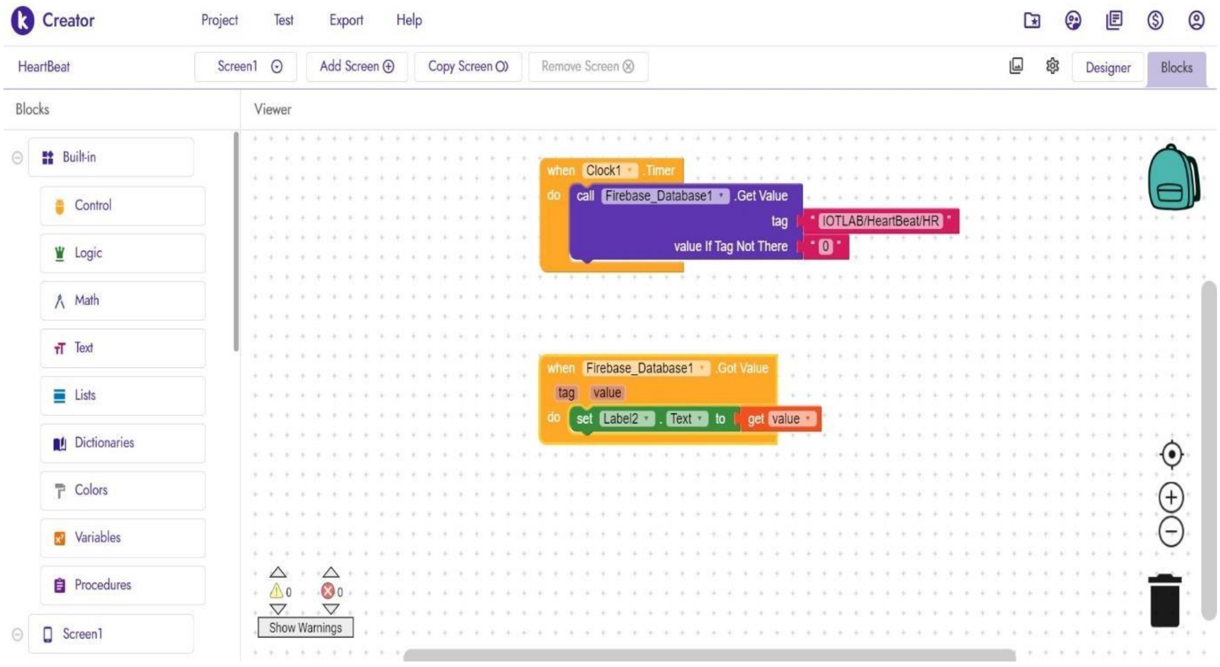
Align Vertical: Top : 1

Background Color: #FFFFFFFF

Background Image: None

Close Screen Animation: Default

Block View:



```
when Clock1 .Timer
do  call Firebase_Database1 .Get Value
                            tag  " IOTLAB/HeartBeat/HR "
            value If Tag Not There  " 0 "

when Firebase_Database1 .Got Value
  tag  value
do  set Label2 . Text to  get value
```

Observations:

Result:

8. Wake Sound Detection and Alarm Notification using IOT

Aim: To detect Wake Sound and Alarm Notification Record sound using a MEMS microphone, transform to frequency domain using FFT, then accordingly detect the presence of signal at a specific frequency of bell and transfer the status to Firebase.

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with INMP441 Sensor and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO VI board to laptop, compile and upload the code to the board.
4. Check BMP values on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to detect the presence of signal at a specific frequency of bell and transfer the status to Firebase

Arduino code:

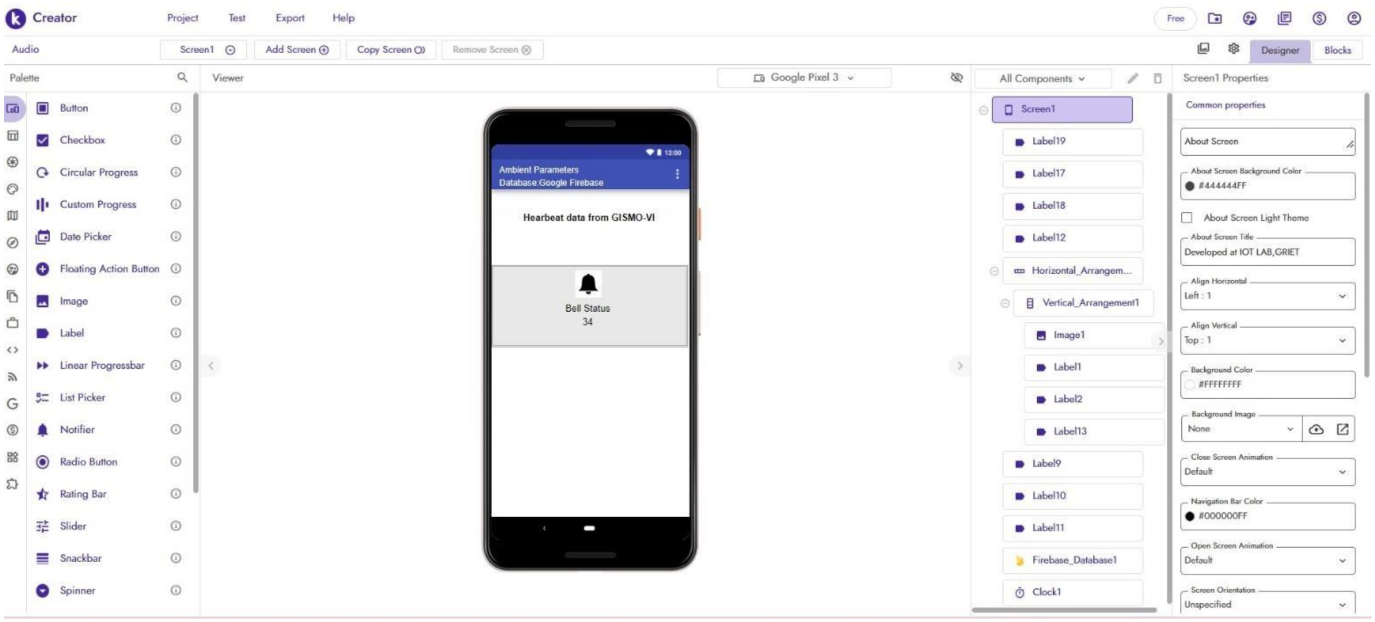
```
#include <Arduino.h>
#include <driver/i2s.h>
#include "arduinoFFT.h"
#include "Credentials.h"
// size of noise sample
#define SAMPLES 1024
const i2s_port_t I2S_PORT = I2S_NUM_0;
const int BLOCK_SIZE = SAMPLES;
#define OCTAVES 9
String tag = "IOTLAB/Audio/BellStatus";
// our FFT data
static float real[SAMPLES];
static float imag[SAMPLES];
static arduinoFFT fft(real, imag, SAMPLES, SAMPLES);
static float energy[OCTAVES];
// A-weighting curve from 31.5 Hz ... 8000 Hz
static const float aweighting[] = {-39.4, -26.2, -16.1, -8.6, -3.2, 0.0, 1.2, 1.0, -1.1};
static unsigned int bell = 0;
static unsigned int fireAlarm = 0;
static unsigned long ts = millis();
static unsigned long last = micros();
static unsigned int sum = 0;
static unsigned int mn = 9999;
static unsigned int mx = 0;
static unsigned int cnt = 0;
static unsigned long lastTrigger[2] = {0, 0};
void OLEDInit();
void OLEDUpdate();
void WiFiinit();
void FirebaseInit();
void FirebaseWrite();
int bellCount;
String bellStatus = " No Bell";
String bellStatus1 = "No_Bell";
static void integerToFloat(int32_t *integer, float *vReal, float *vImag, uint16_t samples)
{
    for (uint16_t i = 0; i < samples; i++)
    {
        vReal[i] = (integer[i] >> 16) / 10.0;
        vImag[i] = 0.0;
    }
}
// calculates energy from Re and Im parts and places it back in the Re part (Im part is zeroed)
static void calculateEnergy(float *vReal, float *vImag, uint16_t samples)
{
    for (uint16_t i = 0; i < samples; i++)
    {
        vReal[i] = sq(vReal[i]) + sq(vImag[i]);
        vImag[i] = 0.0;
    }
}
// sums up energy in bins per octave
static void sumEnergy(const float *bins, float *energies, int bin_size, int num_octaves)
{
    // skip the first bin
    int bin = bin_size;
    for (int octave = 0; octave < num_octaves; octave++)
    {
        float sum = 0.0;
        for (int i = 0; i < bin_size; i++)
        {
            sum += real[bin++];
        }
        energies[octave] = sum;
        bin_size *= 2;
    }
}
static float decibel(float v)
{
    return 10.0 * log(v) / log(10);
```

```
}
// converts energy to logaritmic, returns A-weighted sum
static float calculateLoudness(float *energies, const float *weights, int num_octaves, float
scale)
{
    float sum = 0.0;
    for (int i = 0; i < num_octaves; i++)
    {
        float energy = scale * energies[i];
        sum += energy * pow(10, weights[i] / 10.0);
        energies[i] = decibel(energy);
    }
    return decibel(sum);
}
void setup(void)
{
    Serial.begin(115200);
    Serial.println("Configuring I2S...");
    esp_err_t err;
    // The I2S config as per the example
    const i2s_config_t i2s_config = {
        .mode = i2s_mode_t(I2S_MODE_MASTER | I2S_MODE_RX), // Receive, not transfer
        .sample_rate = 22627,
        .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
        .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT, // for old esp-idf versions use
RIGHT
        .communication_format = i2s_comm_format_t(I2S_COMM_FORMAT_I2S |
I2S_COMM_FORMAT_I2S_MSB),
        .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1, // Interrupt level 1
        .dma_buf_count = 8,                    // number of buffers
        .dma_buf_len = BLOCK_SIZE,             // samples per buffer
        .use_apll = true};
    // The pin config as per the setup
    const i2s_pin_config_t pin_config = {
        .bck_io_num = 14,   // BCKL
        .ws_io_num = 27,    // LRCL
        .data_out_num = -1, // not used (only for speakers)
        .data_in_num = 12   // DOUTESP32-INMP441 wiring
    };
    // Configuring the I2S driver and pins.
    // This function must be called before any I2S driver read/write operations.
    err = i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
    if (err != ESP_OK)
    {
        Serial.printf("Failed installing driver: %d\n", err);
        while (true)
        ;    }
    err = i2s_set_pin(I2S_PORT, &pin_config);
    if (err != ESP_OK)
    {
        Serial.printf("Failed setting pin: %d\n", err);
        while (true)
            ;    }
    Serial.println("I2S driver installed.");
    OLEDInit();
    OLEDUpdate();
    WiFiInit();
    FirebaseInit();
}
unsigned int countSetBits(unsigned int n)
{
    unsigned int count = 0;
    while (n)
    {
        count += n & 1;
        n >>= 1;
    }
    return count;
}
//detecting 2 frequencies. Set wide to true to match the previous and next bin as well
bool detectFrequency(unsigned int *mem, unsigned int minMatch, double peak, unsigned int
bin1, unsigned int bin2, bool wide)
{
    *mem = *mem << 1;
    if (peak == bin1 || peak == bin2 || (wide && (peak == bin1 + 1 || peak == bin1 - 1 || peak ==
bin2 + 1 || peak == bin2 - 1)))
    {
        *mem |= 1;
    }
    if (countSetBits(*mem) >= minMatch)
    {
        return true;
    }
    return false;
}
void sendAlarm(unsigned int index, char *topic, unsigned int timeout)
{
    // do not publish if last trigger was earlier than timeout ms
    if (abs(millis() - lastTrigger[index]) < timeout)
    {
        return;
    }
    lastTrigger[index] = millis();
    //publish to mqtt
    //publish(topic, "1");
}
void sendMetrics(char * topic, unsigned int mn, unsigned int mx, unsigned int avg)
{
    String payload = "{\"min\": ";
    payload += mn;
    payload += ", \"max\":";
    payload += mx;
    payload += ", \"value\":";
    payload += avg;
    payload += "}";
    Serial.println(payload);
    //publish to mqtt
    //publish(topic, (char *)payload.c_str());
}
void calculateMetrics(int val) {
  cnt++;
  sum += val;
  if (val > mx)
  {
      mx = val;
  }
  if (val < mn)
  {
      mn = val;
  }
}

void loop(void)
{
    if (micros() - last < 45200) {
      // send mqtt metrics every 10s while waiting and no trigger is detected
      if (millis() - ts >= 10000 && bell == 0 && fireAlarm == 0)
      {
          //Serial.println(cnt[0]);
          sendMetrics("home/noise", mn, mx, sum / cnt);
          cnt = 0;
          sum = 0;
          mn = 9999;
          mx = 0;
          ts = millis();
         // display.clear();
      // display.display();
      bellStatus = "No Bell";
      bellStatus1 = "No_Bell";
      OLEDUpdate();
      FirebaseWrite();
      Serial.println(bellStatus);
    }
   return;
  }
  last = micros();
     static int32_t samples[BLOCK_SIZE];
  // Read multiple samples at once and calculate the sound pressure
  size_t num_bytes_read;
  esp_err_t err = i2s_read(I2S_PORT,
                          (char *)samples,
                          BLOCK_SIZE, // the doc says bytes, but its elements.
                          &num_bytes_read,
                          portMAX_DELAY); // no timeout
  int samples_read = num_bytes_read / 8;
  // integer to float
  integerToFloat(samples, real, imag, SAMPLES);
  // apply flat top window, optimal for energy calculations
  fft.Windowing(FFT_WIN_TYP_FLT_TOP, FFT_FORWARD);
  fft.Compute(FFT_FORWARD);
  // calculate energy in each bin
  calculateEnergy(real, imag, SAMPLES);
  // sum up energy in bin for each octave
  sumEnergy(real, energy, 1, OCTAVES);
  // calculate loudness per octave + A weighted loudness
  float loudness = calculateLoudness(energy, aweighting, OCTAVES, 1.0);
  unsigned int peak = (int)floor(fft.MajorPeak());
  //Serial.println(peak);
     // detecting 1kHz and 1.5kHz 738Hz
  if (detectFrequency(&bell, 15, peak, 33, 34, true))
  {
     Serial.println("Detected bell");
     sendAlarm(0, "home/alarm/doorbell", 2000);
     bellCount++;
     if(bellCount > 10){
       bellStatus = "Bell !!!!";
       bellStatus1 = "Bell_!_!";
       OLEDUpdate();
       FirebaseWrite();
       Serial.println(bellStatus);
       bellCount = 0;
```

```
        }

    }
        //detecting frequencies around 3kHz
    if (detectFrequency(&fireAlarm, 15, peak, 140, 141, true))
    {
        Serial.println("Detected fire alarm");
        sendAlarm(1, "home/alarm/fire", 10000);
    }
    calculateMetrics(loudness);
}
void OLEDInit(){
  display.init();
  display.setFont(ArialMT_Plain_24);
}
void OLEDUpdate(){
  display.clear();
  display.drawString(0,0,bellStatus);
// display.drawString(40,30,String(beatAvg));
  display.display();
}
void WiFiInit(){
  pinMode(2,OUTPUT);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    digitalWrite(2,!digitalRead(2));
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

}
void FirebaseInit(){
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);

}
void FirebaseWrite(){
  Firebase.setString(firebaseData,tag,bellStatus1);
 }
```

Mobile App:

Designer View:



Ambient Parameters Database:Google Firebase

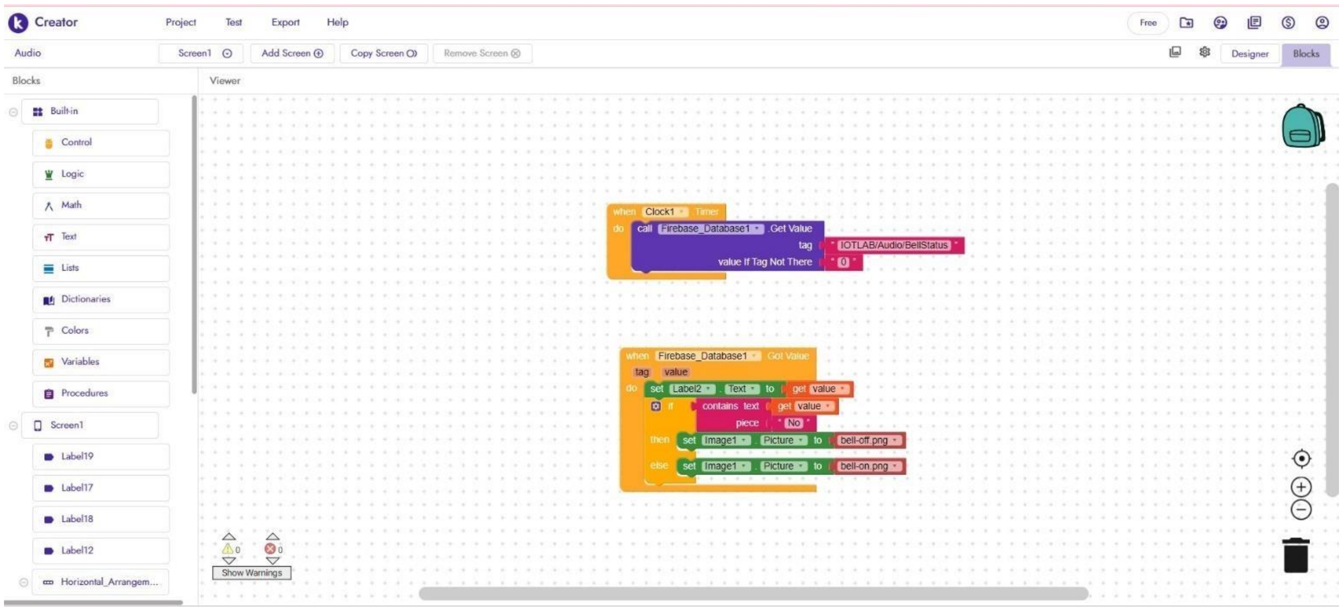
Hearbeat data from GISMO-VI

Bell Status
34

Screen1 Properties

About Screen Background Color: #444444FF
About Screen Title: Developed at IOT LAB,GRIET
Align Horizontal: Left : 1
Align Vertical: Top : 1
Background Color: #FFFFFFFF
Background Image: None
Close Screen Animation: Default
Navigation Bar Color: #000000FF
Open Screen Animation: Default
Screen Orientation: Unspecified

Block View:



```
when Clock1 .Timer
do  call Firebase_Database1 .Get Value
                                tag  "IOTLAB/Audio/BellStatus"
           value If Tag Not There  "0"

when Firebase_Database1 .Got Value
  tag  value
do  set Label2 . Text to  get value
    if    contains text  get value
                  piece  "No"
    then  set Image1 . Picture to  bell-off.png
    else  set Image1 . Picture to  bell-on.png
```

Observations:

Result:

9 IOT based Real Time Appliance Control

Aim: To control the Real Time Appliance based on start, stop timings set in mobile app to update the values in Firebase.

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with relay, DS3231 RTC module and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO VI board with DS3231 RTC module and a relay to laptop, compile and upload the code to the board.
4. Check relay status on serial monitor and OLED display
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to set start and stop time.

Arduino Code:

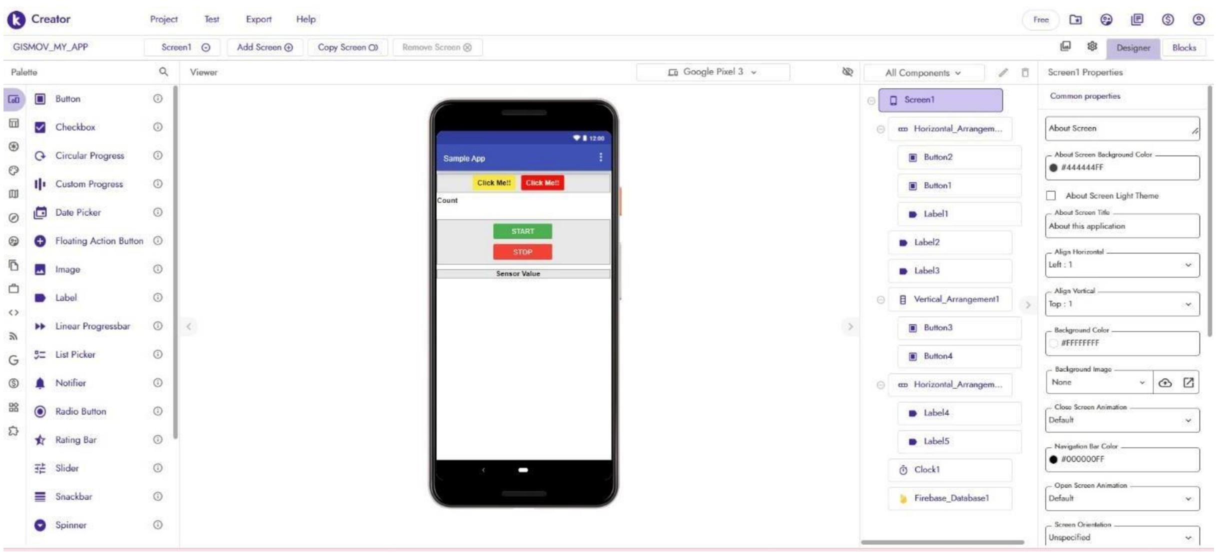
```
#include <Wire.h>
#include "RTClib.h"
RTC_DS3231 rtc;
#include  "SSD1306.h"
SSD1306  display(0x3c, 21, 22);
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"};
#include "FirebaseESP32.h"
#define FIREBASE_HOST "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Do not
include https:// in FIREBASE_HOST
#define FIREBASE_AUTH "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
#define WIFI_SSID "xxxxxxxxxxxxxxxxx"
#define WIFI_PASSWORD "xxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
//Define Firebase Data object
FirebaseData firebaseData;
#define relayPin 13

int delayTime = 3000;

void setup() {
 // put your setup code here, to run once:
pinMode(relayPin,OUTPUT);
Serial.begin(115200);
if(!rtc.begin()){
Serial.print("No RTC module found");
```

```
 while(1);
 }
   //Connecting to Wi-Fi network
 WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
 Serial.print("Connecting to Wi-Fi");
 while (WiFi.status() != WL_CONNECTED)
 {
 Serial.print(".");
 delay(300);
 }
 Serial.println();
 Serial.print("Connected with IP: ");
 Serial.println(WiFi.localIP());
 Serial.println();
 Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
 Firebase.reconnectWiFi(true);

   rtc.adjust(DateTime(F(_DATE_), F(_TIME_)));

 display.init();
 display.setFont(ArialMT_Plain_16);

 }

 void loop() {
 // put your main code here, to run repeatedly:
 display.clear();
 char buffer[10];
 DateTime now = rtc.now();
/* Serial.print(now.year(), DEC);
 Serial.print('/');
 Serial.print(now.month(), DEC);
 Serial.print('/');
 Serial.print(now.day(), DEC);
 Serial.print(" (");
 Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
 Serial.print(") ");
 Serial.print(now.hour(), DEC);
 Serial.print(':');
 Serial.print(now.minute(), DEC);
 Serial.print(':');
 Serial.print(now.second(), DEC);
 Serial.println();*/
 sprintf(buffer,"%02d-%02d-%02d",now.hour(),now.minute(),now.second());
 Serial.println(buffer);

 Firebase.setString(firebaseData,"IOTLAB/Appliance_Control/Current_Time",buffer);

 display.drawString(0,0,buffer);
 int currentTimeTotal;
 currentTimeTotal = now.hour()*60 + now.minute();
 Serial.println(currentTimeTotal);
 String startTimeFull;
 String startTime;
 int startTimeHrs,startTimeMins,startTimeTotal;
 Firebase.getString(firebaseData,"IOTLAB/Appliance_Control/Start_Time",startTimeFull);
 startTime=startTimeFull.substring(2,startTimeFull.length()-2);
 //Serial.println(startTime);
 display.drawString(0,30,startTime); startTimeHrs
 = startTime.substring(0,2).toInt();
 //Serial.println(startTimeHrs);
 startTimeMins = startTime.substring(3).toInt();
 //Serial.println(startTimeMins);
 startTimeTotal=startTimeHrs*60 + startTimeMins;
 Serial.println(startTimeTotal);
 String endTimeFull;
 String endTime;
 int endTimeHrs,endTimeMins,endTimeTotal;
 Firebase.getString(firebaseData,"IOTLAB/Appliance_Control/End_Time",endTimeFull);
 endTime=endTimeFull.substring(2,endTimeFull.length()-2);
 //Serial.println(endTime);
 display.drawString(80,30,endTime); endTimeHrs
 = endTime.substring(0,2).toInt();
 //Serial.println(endTimeHrs);
 endTimeMins = endTime.substring(3).toInt();
 //Serial.println(endTimeMins);
 endTimeTotal=endTimeHrs*60 + endTimeMins;
 Serial.println(endTimeTotal);
if((currentTimeTotal >= startTimeTotal) && (currentTimeTotal < endTimeTotal)){
 digitalWrite(relayPin,HIGH);
 Serial.println("Appliance ON");
 display.setColor(WHITE);
 display.drawCircle(60, 36, 8);
 display.fillCircle(60,36,8);
 Firebase.setInt(firebaseData,"IOTLAB/Appliance_Control/Appliance_Status",1);
 }
Else
{ digitalWrite(relayPin,LOW);
 Serial.println("Appliance OFF");
 display.setColor(BLACK);
 display.drawCircle(60, 36, 8);
 display.fillCircle(60,36,8);
 display.setColor(WHITE);
 display.drawCircle(60, 36, 8);
 Firebase.setInt(firebaseData,"IOTLAB/Appliance_Control/Appliance_Status",0);
 }
```

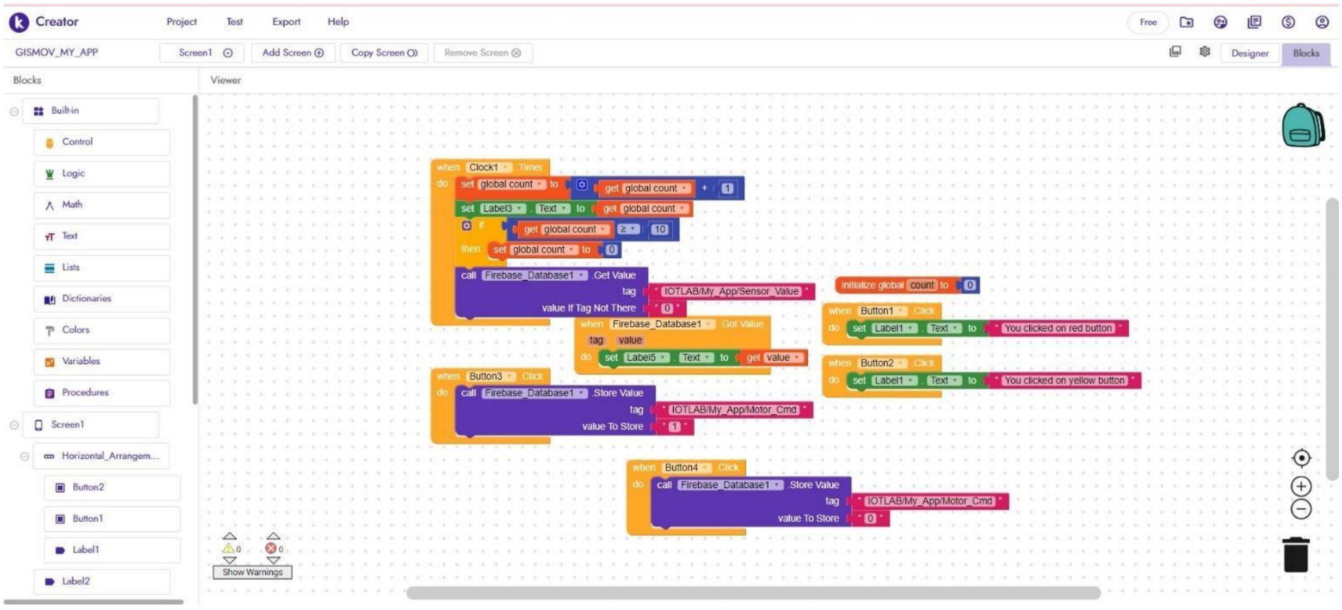
```
display.display();
delay(delayTime);

}
```

Mobile App:

Designer View:



Block View:



```
initialize global count to 0

when Clock1.Timer
do  set global count to get global count + 1
    set Label3 . Text to get global count
    if get global count ≥ 10
    then set global count to 0
    call Firebase_Database1 .Get Value
                            tag "IOTLAB/My_App/Sensor_Value"
              value If Tag Not There "0"

when Firebase_Database1 .Got Value
  tag  value
do  set Label5 . Text to get value

when Button1 .Click
do  set Label1 . Text to "You clicked on red button"

when Button2 .Click
do  set Label1 . Text to "You clicked on yellow button"

when Button3 .Click
do  call Firebase_Database1 .Store Value
                            tag "IOTLAB/My_App/Motor_Cmd"
                  value To Store "1"

when Button4 .Click
do  call Firebase_Database1 .Store Value
                            tag "IOTLAB/My_App/Motor_Cmd"
                  value To Store "0"
```

Observations:

Result:

10. Environment Monitoring using IoT

Aim: To monitor the environment measures such as temperature, atmospheric pressure and altitude using BMP280 sensor, then accordingly transfer data to Firebase and display values in mobile app.

Equipment Required:

1. Laptop with Wi-Fi connection
2. GISMO VI board with BMP280 Sensor and OLED display.
3. Mobile Phone with Kodular Companion App
4. Google Firebase Account

Procedure:

1. Open Arduino IDE, create a new file and write the code
2. Give Wi-Fi and firebase credentials in the code.
3. Connect GISMO VI board to laptop, compile and upload the code to the board.
4. Check BMP values on serial monitor, OLED display and firebase.
5. Design a mobile app with firebase component
6. Test and download the mobile app using Kodular Companion to monitor the environment measures such as temperature, atmospheric pressure and altitude.

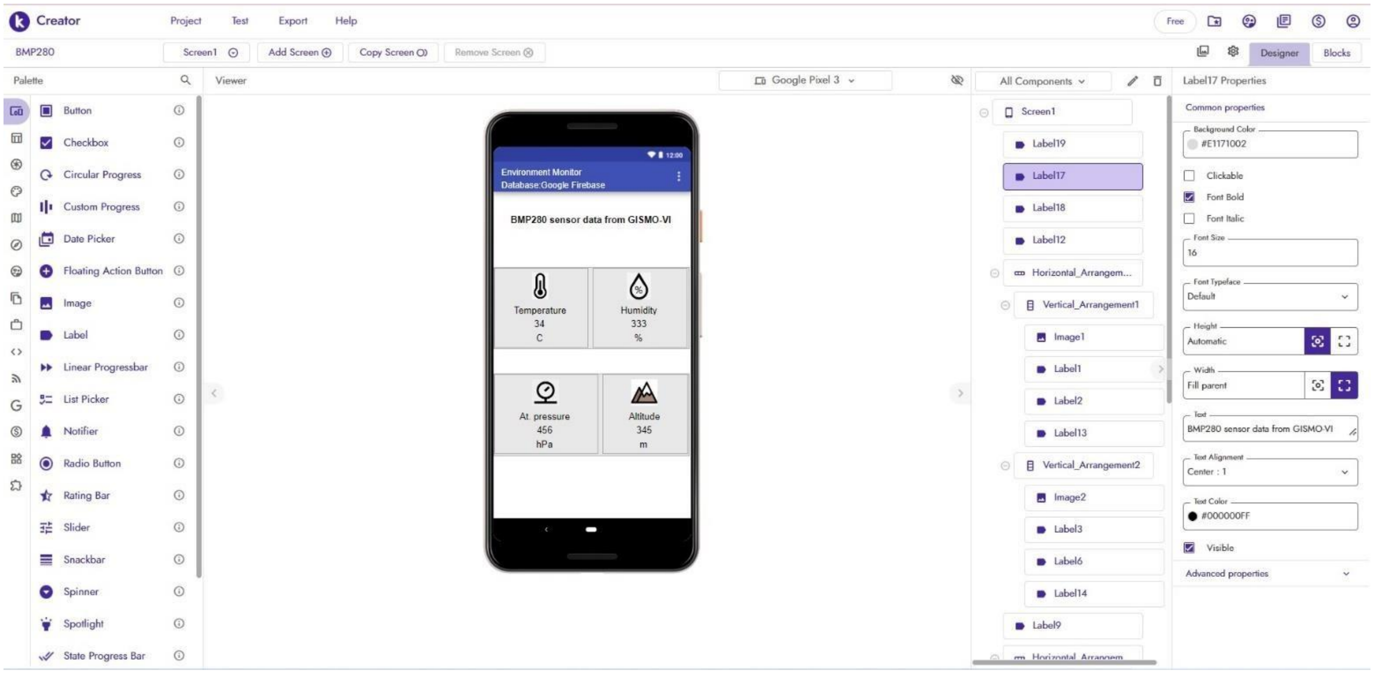
Arduino Code:

```
#include <Wire.h>
#include <Adafruit_BMP280.h>

Adafruit_BMP280 bmp; // I2C
#include "Credentials.h"

#define SEALEVELPRESSURE_HPA (1013.25)

float tempC,tempF,atPressure,altitude,humidity;
void setup() {
  Serial.begin(115200);
  Serial.println(F("BMP280 test"));

  if (!bmp.begin(BMP280_ADDRESS_ALT, BMP280_CHIPID)) {
      Serial.println(F("Could not find a valid BMP280 sensor, check wiring or "
                   "try a different address!"));
    while (1) delay(10);
  }

  /* Default settings from datasheet. */
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode. */
                 Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
                 Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
                 Adafruit_BMP280::FILTER_X16,      /* Filtering. */
                 Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */

 OLEDInit();
 WiFiInit();
 FirebaseInit();
}
```

```
void loop() {
    printValues();
    displayValues();
    delay(2000);
}

 void OLEDInit(){
  display.init();
  display.setFont(ArialMT_Plain_16);
}

void printValues(){
 // float tempC,tempF,atPressure,altitude,humidity;
  Serial.print("Temperature = ");
  tempC = bmp.readTemperature();
//Serial.print(bme.readTemperature());
Serial.print(tempC);
Serial.println(" deg C");
Firebase.setFloat(firebaseData,"IOTLAB/Environment_Monitor/Temperature",tempC);

Serial.print("Temperature = ");
//Serial.print(1.8*bme.readTemperature()+32);
Serial.print(1.8*tempC+32);
Serial.println(" deg F");

Serial.print("Pressure = ");
atPressure=bmp.readPressure()/100.0F;
//Serial.print(bme.readPressure()/100.0F);
Serial.print(atPressure);
Serial.println(" hPa");
Firebase.setFloat(firebaseData,"IOTLAB/Environment_Monitor/Pressure",atPressure);

Serial.print("Altitude = ");
altitude=bmp.readAltitude(SEALEVELPRESSURE_HPA);
//Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.print(altitude);
Serial.println(" m");
Firebase.setFloat(firebaseData,"IOTLAB/Environment_Monitor/Altitude",altitude);
Serial.println();
Firebase.setFloat(firebaseData,"IOTLAB/Environment_Monitor/Humidity",0.0);

}

void displayValues(){
  display.clear();

  String myString = "";

  char buffer[6];
  dtostrf(tempC,5,1,buffer);
  myString.concat(buffer);
  myString.concat(" C");
  display.drawString(0,0,myString);

  humidity = 0.0;
  myString = "";
  dtostrf(humidity,5,1,buffer);
  myString.concat(buffer);
  myString.concat(" %");
  display.drawString(64,0,myString);

 myString = "";
  dtostrf(atPressure,5,1,buffer);
  myString.concat(buffer);
  display.drawString(0,30,myString);

    myString = "";
  dtostrf(altitude,5,1,buffer);
  myString.concat(buffer);
  myString.concat("m");
  display.drawString(64,30,myString);


  display.display();

}
void WiFiInit(){
  pinMode(2,OUTPUT);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    digitalWrite(2,!digitalRead(2));
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

}
void FirebaseInit(){
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);

}
```

Mobile App:

Designer View:



Environment Monitor
Database:Google Firebase

BMP280 sensor data from GISMO-VI

Temperature
34
C

Humidity
333
%

At. pressure
456
hPa

Altitude
345
m

Label17 Properties

Common properties

Background Color: #E1171002

Clickable ☐
Font Bold ✓
Font Italic ☐

Font Size: 16

Font Typeface: Default

Height: Automatic

Width: Fill parent

Text: BMP280 sensor data from GISMO-VI

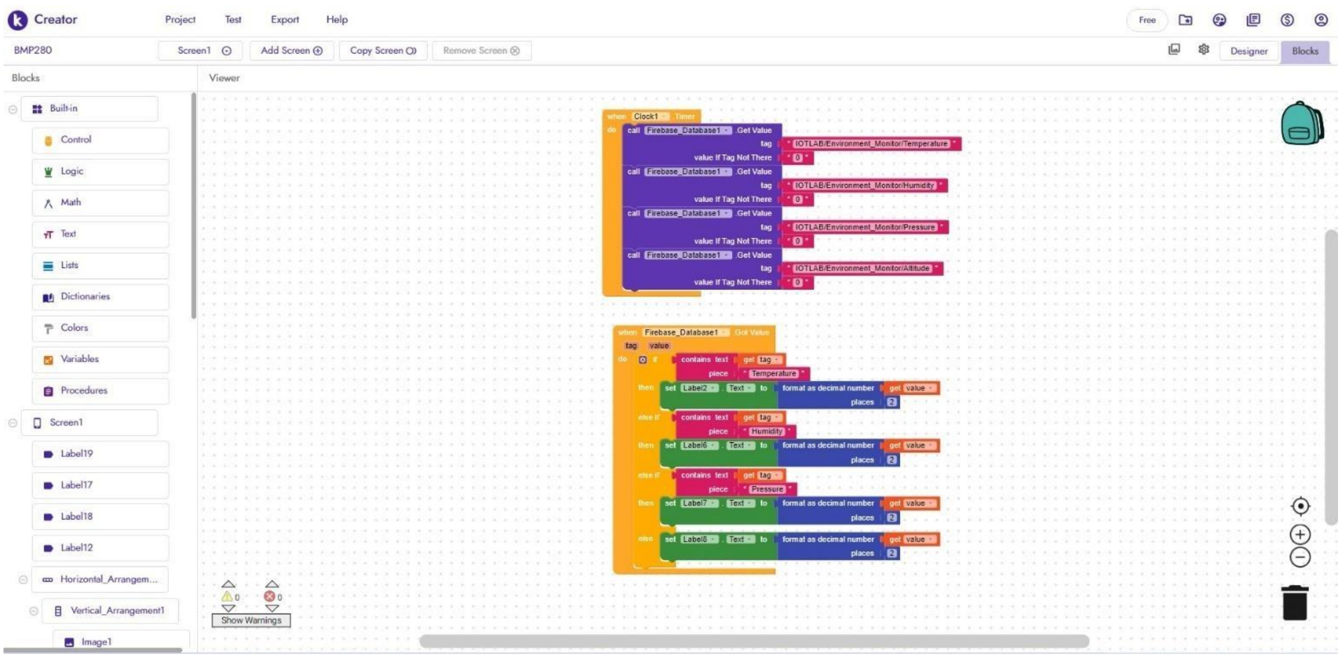
Text Alignment: Center : 1

Text Color: #000000FF

Visible ✓

Advanced properties

Block View:



```
when Clock1 .Timer
do  call Firebase_Database1 .Get Value
                              tag  "IOTLAB/Environment_Monitor/Temperature"
           value If Tag Not There  "0"
    call Firebase_Database1 .Get Value
                              tag  "IOTLAB/Environment_Monitor/Humidity"
           value If Tag Not There  "0"
    call Firebase_Database1 .Get Value
                              tag  "IOTLAB/Environment_Monitor/Pressure"
           value If Tag Not There  "0"
    call Firebase_Database1 .Get Value
                              tag  "IOTLAB/Environment_Monitor/Altitude"
           value If Tag Not There  "0"

when Firebase_Database1 .Got Value
 tag  value
do  if  contains text  get tag
                piece  "Temperature"
    then  set Label2 . Text to  format as decimal number  get value
                                                  places  2
    else if  contains text  get tag
                     piece  "Humidity"
    then  set Label6 . Text to  format as decimal number  get value
                                                  places  2
    else if  contains text  get tag
                     piece  "Pressure"
    then  set Label7 . Text to  format as decimal number  get value
                                                  places  2
    else  set Label8 . Text to  format as decimal number  get value
                                                  places  2
```

Observations:

Result: