Digital I/O Interface

simple LED Blink

```
void setup(){
    pinMode(18, OUTPUT)
}

void loop(){
    digitalWrite(18, HIGH);
    delay(500);
    digitalWrite(18, LOW);
    delay(500);
}
```

IR sensor


```
#define LED 18
#define IR 14

void setup(){
    pinMode(LED, OUTPUT);
    pinMode(IR, INPUT);
}

void loop(){
    bool IRStatus = digitalRead(IR);
    if (IRStatus){
        digitalWrite(LED, HIGH);
    }
    else{
        digitalWrite(LED, LOW);
```

```
    }
    delay(5000);
}
```

serial communication

```
#define LED 9

void setup(){
    Serial.begin(9600);
    pinMode(LED, OUTPUT);
}

void loop(){
    if(Serial.available()){
        char in = Serial.read();
        if(in == 'a'){
            Serial.println("Blinking the LED!");
            digitalWrite(LED, HIGH);
            delay(2000);
            digitalWrite(LED, LOW);
        }
    }
}
```

pulse width modulation

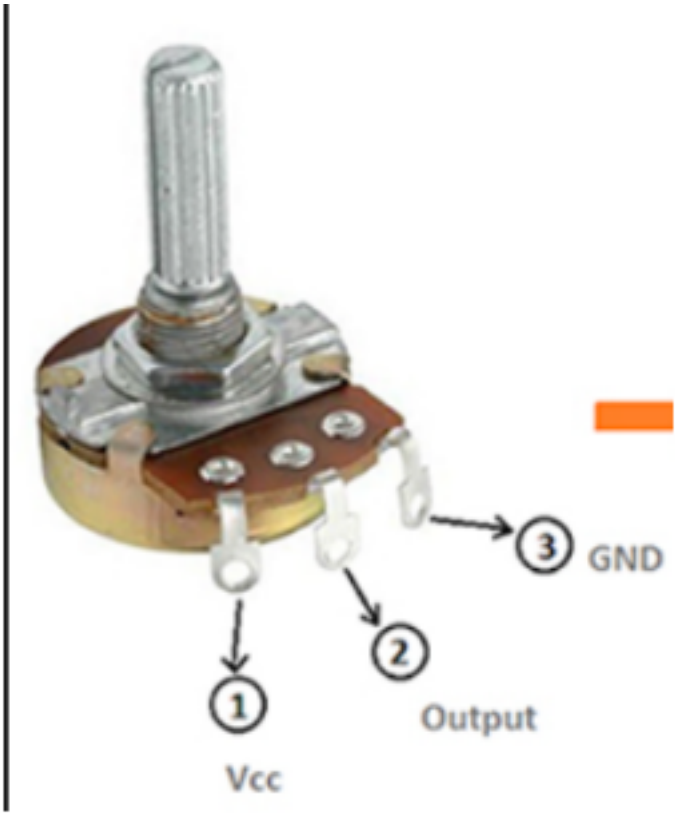
```
#define ledPin 26;
#define freq 5000;
#define ledChannel 0;
#define resolution 8;

void setup(){
  ledcAttachChannel(ledPin, freq, resolution, ledChannel);
}

void loop(){
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
    ledcWrite(ledPin, dutyCycle);
    delay(15);
  }
}
```

Analog Read and Write

potentiometer




```
#define POT 34
#define LED 9

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int sensorValue = analogRead(sensorPin);
  int outputValue = map(sensorValue, 0, 1023, 0, 255);
  Serial.println(sensorValue);
  analogWrite(ledPin, outputValue);
}
```

DHT sensor

```
#include <dht.h>

#define DHTPIN 13
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup(){
    Serial.begin(9600);
    dht.begin();
}

void loop(){
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();
    float heatindex = dht.computeHeadIndex();

    Serial.println("Temperature: "+String(temperature));
    Serial.println("Humidity: "+String(humidity));
    Serial.println("HeatIndex: "+String(heatindex));
}
```

light sensor


```
#define LIGHT 36

void setup(){
    Serial.begin(9600);
}

void loop(){
    int lightValue = analogRead(LIGHT);
    Serial.println("Light: "+String(lightValue));

    if (lightValue < 40){
        Serial.println("Dark");
    }
    else if(lightValue < 800){
        Serial.println("Dim");
    }
    else if(lightValue < 2000){
        Serial.println("Light");
    }
    else if(lightValue < 3200){
        Serial.println("Bright");
    }
    else{
        Serial.println("Very Bright");
    }
}
```

LM35

```
#define ADC_VREF_mV    3300.0
#define ADC_RESOLUTION 4096.0
#define LM35       36

void setup(){
   Serial.begin(9600);
}

void loop(){
   int adcValue = analogRead(LM35);
   float tempC = (adcValue * (ADC_VREF_mV/ADC_RESOLUTION))/ 10;
   Serial.println("Temperature: " + String(adcValue));
   delay(500);
}
```

```
import RPi.GPIO as GPIO

LED = 2
IR = 4

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED, GPIO.OUT)
GPIO.setup(IR, GPIO.IN)

while True:
    if GPIO.input(IR):
        GPIO.output(LED, GPIO.HIGH)
    else:
        GPIO.output(LED, GPIO.LOW)
```

Camera Module Interface using Rpi

1) video record

```
from time import sleep
from picamera2 import PiCamera2, Preview #type: ignore
from picamera2.encoders import H264Encoder #type: ignore

picam2 = PiCamera2()
picam2.configure(picam2.create_video_configuration())
picam2.start_preview(Preview.QTGL)
picam2.start_recording(H264Encoder(10000000), 'myvideo.h264')
sleep(5)
picam2.stop_recording()
picam2.stop_preview()
```

2) image capture

```
from picamera2 import Picamera2, Preview
from time import sleep

picam2 = Picamera2()
picam2.start_preview(Preview.QTGL)

picam2.start()
for i in range(5):
    picam2.start_and_capture_file(f"image_{i+1}.jpeg")
    sleep(2)
picam2.stop_preview()
picam2.close()
```

Implementation of I2C

1) I2C between RPi and Arduino

raspberry pi code:

```
from smbus import SMBus

addr = 0x8
bus = SMBus(1)
print("Enter 1 for ON or 0 for OFF")
while True:
    bus.write_byte(addr, int(input(">>>")))
```

arduino code:

```
#include <Wire.h>

#define LED 13

void setup(){
    Wire.begin(0x8);
    Wire.onReceive(receiveEvent);
    pinMode(LED, OUTPUT);
}

void receiveEvent(int n){
    while (Wire.available()){
        char c = Wire.read();
        digitalWrite(LED, c);
    }
}

void loop() {}
```

Configure and connection establishment of Wi-Fi with IoT development board

1) Wi-Fi communication between pi and ESP32

ESP32 code:

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "wifi_name";
const char* password = "password";

void setup(){
  Wifi.begin(ssid, password);
  while(Wifi.status() != WL_CONNECTED) delay(500);
  Serial.println("Connected Ip: "+String(WiFi.localIP()));
}

void loop(){
  if (WiFi.status == WL_CONNECTED){
    HTTPClient http;
    http.begin('http://ip:port/get-sensor?temperature=123');
    int status = http.GET();
    if(status > 0){
      String response = http.getString();
      Serial.println(response);
    }
    else{
      Serial.println("Error on HTTP request");
    }
    http.end();
  }
  delay(5000);
}
```

Raspberry Pi code:

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/get-sensor', methods=['GET'])
def data():
  if request.method == 'GET':
    print(f"Received: {request.args.get('temperature')}")
    return ('OK', 200)
  else:
    return ('Method Not Allowed!', 405)
```

```
if __name__ == '__main__':
    app.run(debug=True, port=8000, host='0.0.0.0')
```

MQTT – Publish and Subscribe with IoT development board

1) MQTT server and client

Raspberry pi:
mosquitto_sub -d -t ESP32/Temperature

ESP32:

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "wifi_name";
const char* password = "password";

WiFiClient espClient;
PubSubClient client(espClient);

void setup(){
    Wifi.begin(ssid, password);
    while(Wifi.status() != WL_CONNECTED) delay(500);
    Serial.println("Connected Ip: "+String(WiFi.localIP()));

    client.setServer("mqtt_ip", 1883);
    while(!client.connected());
    Serial.println("Connected!");
}

void loop(){
    client.publish("ESP32/Temperature", tempString);
}
```

Task Context Maintenance

1) FreeRTO task scheduling

```
#include <string.h>

TaskHandle_t xHandle1, xHandle2;

void setup() {
    Serial.begin(115200);
    String s1 = "Hello";
    String s2 = "World";
    xTaskCreate(printTask, "printHello", 1000, &s1, 1, &xHandle1);
    xTaskCreate(printTask, "printWorld", 1000, &s2, 2, &xHandle2);
}
void loop() {
    delay(1000);
}

void printTask(void* parameter)
{
    String s = *(String*)parameter;
    for (int i = 0; i < 10; i++){
        Serial.println(s);
        delay(100);
    }
    vTaskDelete(NULL);
}
```

Mutual Exclusion

1) mutual exclusion with semaphore

```
TaskHandle_t xHandle1, xHandle2;
SemaphoreHandle_t semvar;

void setup(){
    Serial.begin(115200);
    if (semvar == NULL){
        semvar = xSemaphoreCreateBinary();
    } else{
        xSemaphoreGive(semvar);
    }
    int LED_1 = 17;
    int LED_2 = 18;
    pinMode(LED_1, OUTPUT);
    pinMode(LED_2, OUTPUT);
    xTaskCreate(blink, "blink LED 1", 10000, &LED_1, 1, &xHandle1);
    xTaskCreate(blink, "blink LED 2", 10000, &LED_2, 2, &xHandle2);
}

void blink(void *param){
    int LED = *(int *)param;
    for (int i = 0; i < 10; i++){
        if (xSemaphoreTake(semvar, (TickType_t) 1) == pdTRUE){
            digitalWrite(LED, HIGH);
            delay(500);
            digitalWrite(LED, LOW);
            delay(500);
            xSemaphoreGive(semvar);
        }
    }
    vTaskDelete(NULL);
}
```

Import Sensor data into cloud database

1) Thing Speak

```
#include <WiFi.h>
#include "ThingSpeak.h"

#define PIN_LM35 35

const char* ssid = "xxxxx";
const char* password = "yyyyyy";
WiFiClient client;
unsigned long myChannelNumber = zzzzzz;
const char *myWriteAPIKey = "mmmmmmmmmmmm";

void setup()
{
    pinMode(PIN_LM35, INPUT);
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    ThingSpeak.begin(client);
}

void loop() {
    if(WiFi.status() != WL_CONNECTED){
        Serial.print("Attempting to connect");
        while(WiFi.status() != WL_CONNECTED){
            WiFi.begin(ssid, password);
            delay(5000);
        }
        Serial.println("\nConnected.");
    }
    int adcVal = analogRead(PIN_LM35);
    float tempC = (adcVal * (3300 / 4096)) / 10;
    Serial.println("Temperature: " + String(tempC));
    int x = ThingSpeak.writeField(myChannelNumber, 1, tempC, myWriteAPIKey);
    if(x == 200){
        Serial.println("Channel update successful.");
    } else{
        Serial.println("Problem updating channel. HTTP error code " + String(x));
    }
    delay(5000);
}
```