Q1. Design a simple circuit diagram for an IoT temperature sensor node using an Arduino microcontroller, a

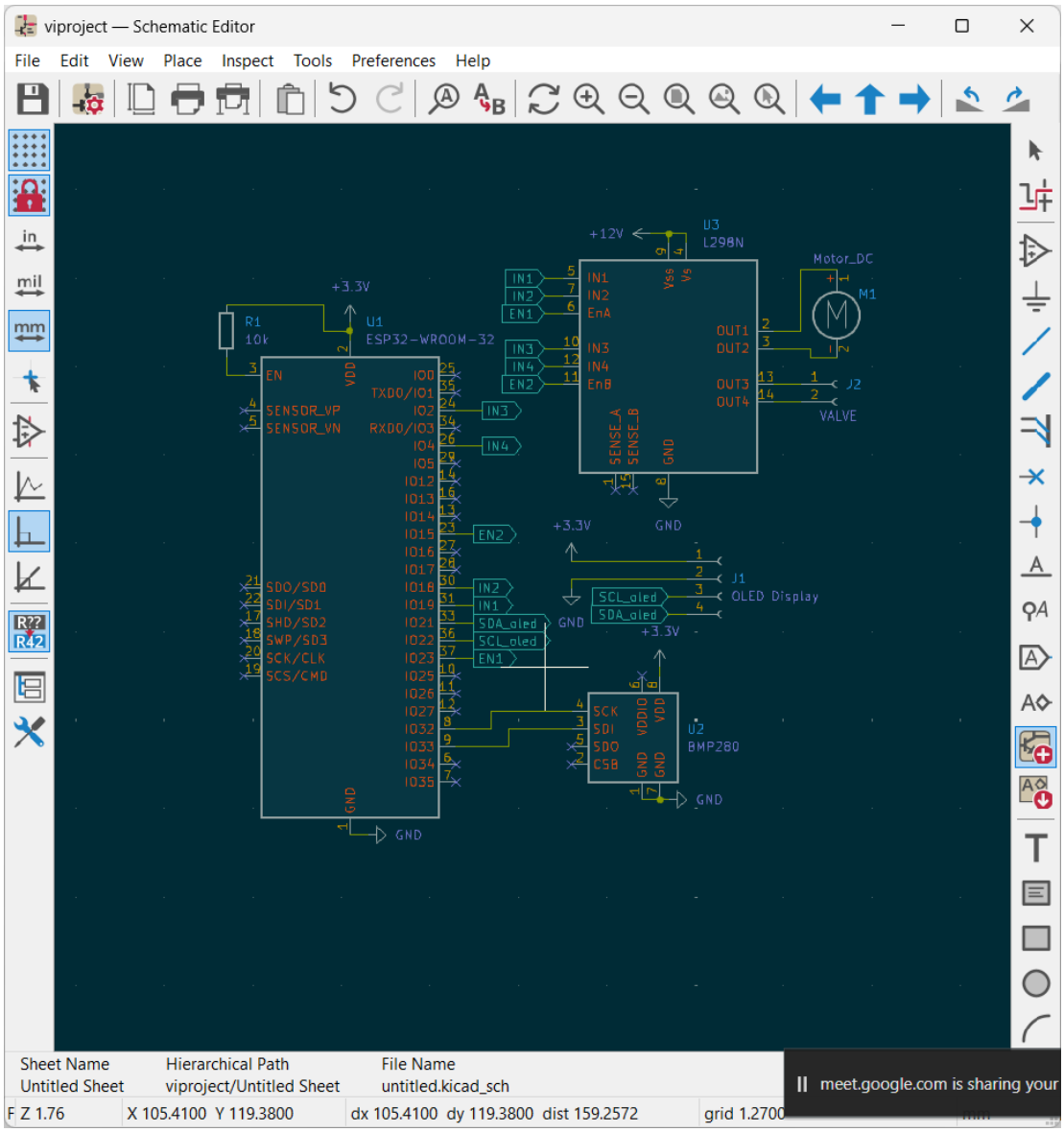
temperature sensor (e.g., LM35), and a Wi-Fi module (e.g., ESP8266).Provide the connections and

components required.

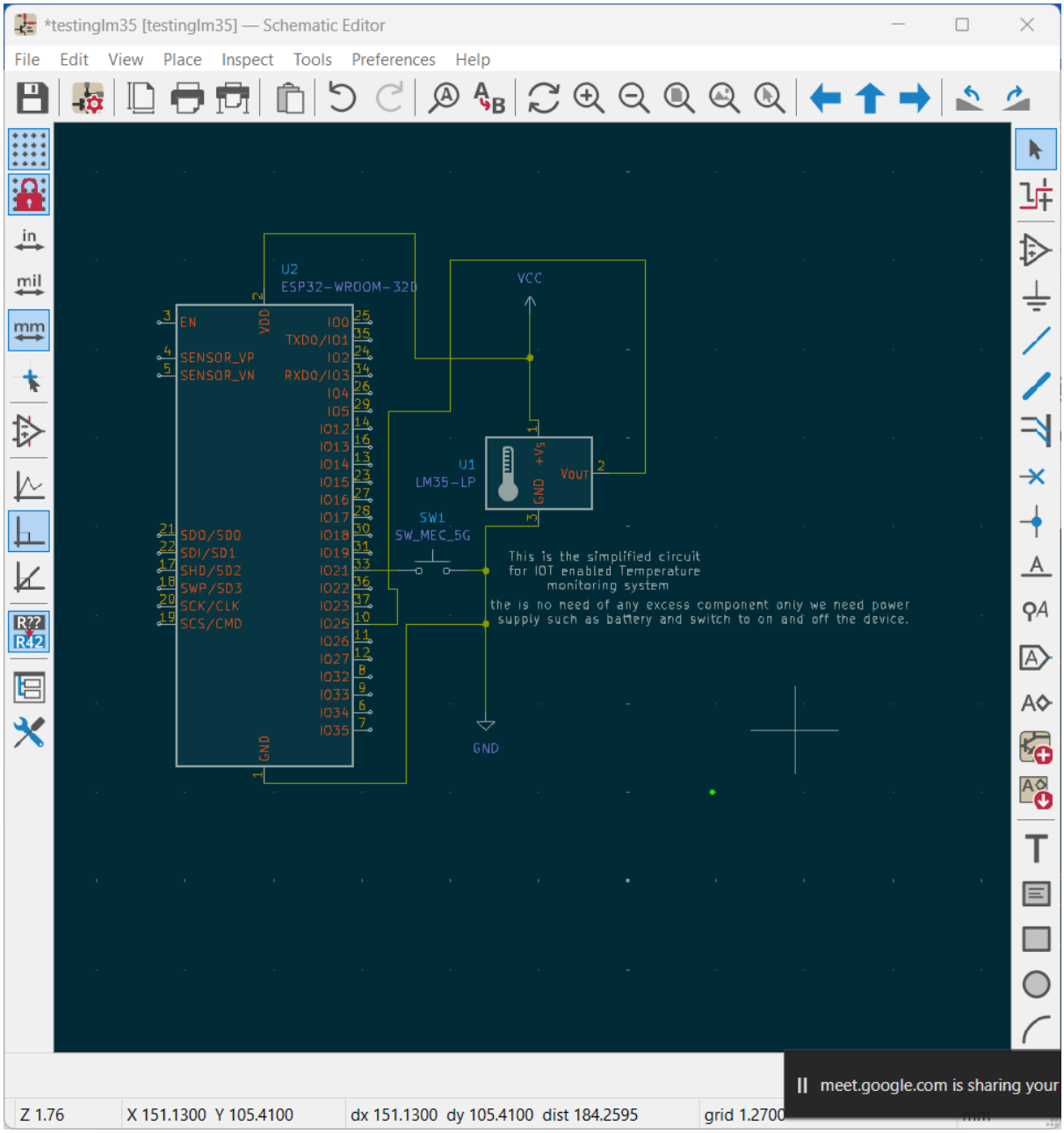
Ans:

Prototype 1.

instead of LM35 I Have used BMP280 which monitor temperature and pressure and few modules like motor driver to make working with motor. Also I've used esp32 for Bluetooth and wifi connectivity. To get real time reading Ive added 128x64 pixel display



Prototype 2:



Que 2. Explain the purpose and functionality of pull-up and pull-down resistors in digital circuits.How are they typically used in embedded systems?

Ans:

Pull up and pull down resistor are basically use to make signal strength good enough for working with microcontroller and sensors. Pull up resistor is basically connected directly with Vcc and other terminal of resistor is connected to signal terminal of microcontroller, similarly pull down resistor is direcly connected to GND ground. And other terminal is connected to microcontroller.

This is external pull up and pull down scenario, but in general few microcontroller and sensor modules come with internal pull up resistor. Usually we use pull up resistor wit I2c interfacing.

In above Que 1 Ive used switch with is pull down configuration.

Que 3. Describe the differences between UART, SPI, and I2C communication protocols. When wouldyou choose one over the others in a hardware design?

Ans:

UART is universal asynchronous transreceiver interface protocol which is not use clock signal, I mean not use PWM signal to work with specifics frequency. It is use in generall serial communication.

It has RX and TX pins

All mentioned protocol are serial interfaces.

SPI is serial peripheral interface. I has MOSI, MISO and CS chip select and Clock pins there is requirement of clock signal to work AS DATA PACKETS ARE USE THIS TO COMMINICATE .

SPI IS VARY FASTER THEN i2C.

I2C IS INTER INTIGRATED CIRCUIT IT USED IN EMBEDDED SYSTEM VIDELY. THIS COMES WITH MINIUM SPEED AND DATA TRANSFER RATE BUT AROUND 20KBPS TO 1MBPS. I2C HAVE VARIOUS UPDATED VERSION.

SPI IS USE TO CONTROL VARIOUS SLAVE AND USING MASTER. SIMILARLY I2C ALSO USE TO COMMUNICATE WITH SLAVE USING ADRESSES OF SLAVE.

IN SPEED COMPARIASUN UART >SPI >I2C. I AM NOT SHORE ABOUT UART BUT VARIOUS DIVICES USE UART AS WE KNOW USB IS A TYPE OF UART IT USE d+ AND d- FOR COMMUNICATION.

QUE 4. Write a C function to read data from an I2C temperature sensor (e.g., TMP102) connected toan Arduino microcontroller. Assume the Arduino is configured as the master device.

ANS: BELOW MEENTIONED CODE IS SIMPLE CODE SNIPPED FOR PROTOTYPE 2, IT READ THE gpio ONLY.

```
// i've used above for making this code i haven't add gpio.h header, i've
directly mapped resisters.
#include <stdint.h>
#include <stdio.h>

#define GPIO_INPUT_PIN 25 // Define GPIO pin number

// Register addresses
#define GPIO_ENABLE_REG 0x3FF44020  // GPIO enable register
#define GPIO_OUT_REG    0x3FF44004  // GPIO output register
#define GPIO_IN_REG     0x3FF4403C  // GPIO input register           --- Added
for future use
```

```
#define GPIO_PIN_MUX_REG 0x3FF49024 // Pin multiplexing register    --- Added
for future use

// Function to set a GPIO pin as input
void gpio_set_direction(uint8_t gpio_num, uint8_t mode) {
    if (mode) { // If mode is 1, set as output
        *(volatile uint32_t *)(GPIO_ENABLE_REG) |= (1 << gpio_num);
    } else { // Set as input if 0 is placed
        *(volatile uint32_t *)(GPIO_ENABLE_REG) &= ~(1 << gpio_num);
    }
}

// Function to read the level of a GPIO pin
uint8_t gpio_get_level(uint8_t gpio_num) {
    return (*(volatile uint32_t *)(GPIO_IN_REG) >> gpio_num) & 0x01; // & 0x01
is used to isolate the least significant bit (LSB)
}                                                                    // of the
value read from the GPIO input register, mask all, (to extract a single bit
from a multi-bit value)

void main() {
    // Set GPIO_INPUT_PIN as input
    gpio_set_direction(GPIO_INPUT_PIN, 0);

    while (1) {
        // Read the level of the input pin
        uint8_t level = gpio_get_level(GPIO_INPUT_PIN);

        // Print the level to Terminal or serial monitor. same time connect
FFT Analyzer
        printf("GPIO %d Level: %d\n", GPIO_INPUT_PIN, level);

        // Add a delay if necessary (e.g., using vTaskDelay)
    }
}
```

HERE IS THE ARDUINO CODE TO INTERFACE WITH TEMPERATURE SENSOR I2C AND OLED I2C

```
#include <Wire.h>
#include <Adafruit_BMP280.h>
#include <U8g2lib.h>
#include <BLEDevice.h>
Adafruit_BMP280 bmp(&Wire1); // CUSTOM PINS

// Initialize the u8g2 library for a 128x64 I2C OLED using the default I2C bus
(Wire)
```

```
U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE, /*
clock=*/ 22, /* data=*/ 21);
// BLE Variables
BLEServer* pServer = NULL;
BLECharacteristic* pCharacteristic = NULL;
String sensorData;  // For BLE transmission of sensor data
oid setup() {
    Serial.begin(115200);

    // Initialize I2C buses and BMP280
    Wire.begin();
    Wire.setClock(100000);
    Wire1.begin(33, 32);
    Wire1.setClock(100000);
      // Initialize BMP280 sensor
    if (!bmp.begin(0x76)) {
        u8g2.clearBuffer();
        u8g2.setFont(u8g2_font_ncenB08_tr);
        u8g2.drawStr(5, 10, "No BMP280 sensor!");
        u8g2.sendBuffer();
        while (1) delay(10);
    }
        // Set BMP280 sensor parameters
    bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
Adafruit_BMP280::SAMPLING_X2, Adafruit_BMP280::SAMPLING_X16,
Adafruit_BMP280::FILTER_X16, Adafruit_BMP280::STANDBY_MS_500);

    // Initialize BLE
    BLEDevice::init("BTAG");
    pServer = BLEDevice::createServer();
    BLEService* pService = pServer->createService("4fafc201-1fb5-459e-8fcc-
c5c9c331914b");
    pCharacteristic = pService->createCharacteristic("beb5483e-36e1-4688-b7f5-
ea07361b26a8", BLECharacteristic::PROPERTY_READ |
BLECharacteristic::PROPERTY_NOTIFY);
    pService->start();

    BLEAdvertising* pAdvertising = BLEDevice::getAdvertising();
    pAdvertising->addServiceUUID(pService->getUUID());
    pAdvertising->setScanResponse(true);
    pAdvertising->setMinPreferred(0x06);
    pAdvertising->setMinPreferred(0x12);
    BLEDevice::startAdvertising();

    // Initialize OLED display
    u8g2.begin();
    u8g2.clearBuffer();
    }
```

```
    // Task for reading sensor data and BLE communication
void sensorTask() {
    float temperature = bmp.readTemperature();
    float calibratedPressure = bmp.readPressure() - 83668;

    // Prepare sensor data to send via BLE
    sensorData = "Temp: " + String(temperature) + " C, Pressure: " +
String(calibratedPressure) + " Pa ";
    pCharacteristic->setValue(sensorData.c_str());
    pCharacteristic->notify();
}
// Task for updating the OLED display
void displayTask() {
    u8g2.clearBuffer();
    u8g2.setFont(u8g2_font_ncenB08_tr);
    u8g2.setCursor(20, 15);
    u8g2.print("Temp: ");
    u8g2.print(bmp.readTemperature());
    u8g2.print(" C");
    u8g2.setCursor(0, 30);
    u8g2.print("Pressure: ");
    u8g2.print(bmp.readPressure() - 83668);
    u8g2.print(" Pa");
    u8g2.sendBuffer();
}
void loop() {
    sensorTask();
    displayTask();
}
```

Que 5. Explain the concept of interrupt-driven programming in embedded systems. How would youuse interrupts to handle a button press event in an Arduino sketch?

Ans:

The is a crucial and important task to handle interrupt and service routine. To make interrupt driven system we need to know about Finite state machine and COntroll system.

As example if we like to make switch driven simple project we need to use switch as interrupt, it may be pull up or pull down as per GPIO pin assign to microcontroller i.e assigning pin OUTPUT or input as low or high. 0 or 1 I.e internal pull up or pull down vice versa.

Led blinking with switch is a good example.

Int switch = 10;

Int led =13;

Pinmode (10,input);

Pinmode (13, output);

Then

Writing digital or analog write function we can define led on off state using if condition.

Que 6. Describe the process of flashing firmware onto a microcontroller using a tool like AVRDUDE or Arduino IDE. What are the steps involved, and what precautions should be taken?

Ans;

Arduino ide is great combination of compiler, linker and debugger. Also it has bootloader capability to dump firmware on controller there are various controller which are now supported on IDE. So it is wonderful tool.

Step1. Make sure that your computer have proper driver install to communicate with bootloader

2. write code and verify using verify bottun if you see your board is not connected or not selected make sure to select other wise code will not dump.

3. to upload there is upload button.

Few things to make sure like inside tools. U can see various important things. Board, port and manage library.

Que 8. Given a scenario where an IoT device intermittently loses connection to the Wi-Fi network, outline the steps

you would take to diagnose and troubleshoot the issue. What potential hardware or firmware-related problems could cause this issue?

Ans:

In Hardware I will see the proper power supply connection as Antenna is power hungry part of microcontroller so I've check 5v supply is connected to controller or not propper decoupling capacitor is connected or not. Connection is proper or not , is there any interference present or not? I will check this thing.

In firmware I use service routing call function if wifi or Bluetooth is not connected then I will all that function till it connected.

Que 9.

Develop a simple firmware algorithm to implement a low-power sleep mode in an embedded system. Describe how you would optimize power consumption while ensuring timely wake- up for sensor readings or communication tasks.

Ans:

I will set the microcontroller in interrupt service routing mode. So that is will start working when there is any interrupt found.

There are mode like deep sleep, where controller is all time in off state except watch dog timer. Which is counting on state of interrupt. There is NVIS in STM32 which is use to make this thing happen.

We can use while loop or switch case to make logic. As per switch press.

```
While (Switch == HIGH){


Void sleep();

}

//sleep function

void sleep() { //design the firmware for deep sleep mode

ESP.deepSleep(0);

}
```

This is the function in Arduino library which we can call during interrupt handling.

It will redure the unnecessary power consumption in battery operated devices.