TOPST VCP-G Arduino Sensor

User Guide

Rev. 1.01 [G]
2025-07-18

TABLE OF CONTENTS

Contents

Figures

Tables

1 INTRODUCTION

This document provides guidelines on using various Arduino sensors with the VCP-G board. It includes connection instructions and example codes to help you develop projects easily by using the VCP-G board.

Specifically, this document provides guidance on Arduino IDE examples for the VCP-G board, including:

- VCP-G Digital
- VCP-G Analog
- VCP-G SPI
- VCP-G I2C
- VCP-G UART
- Additional Example

Refer to Figure 1.1 before using the VCP-G board.

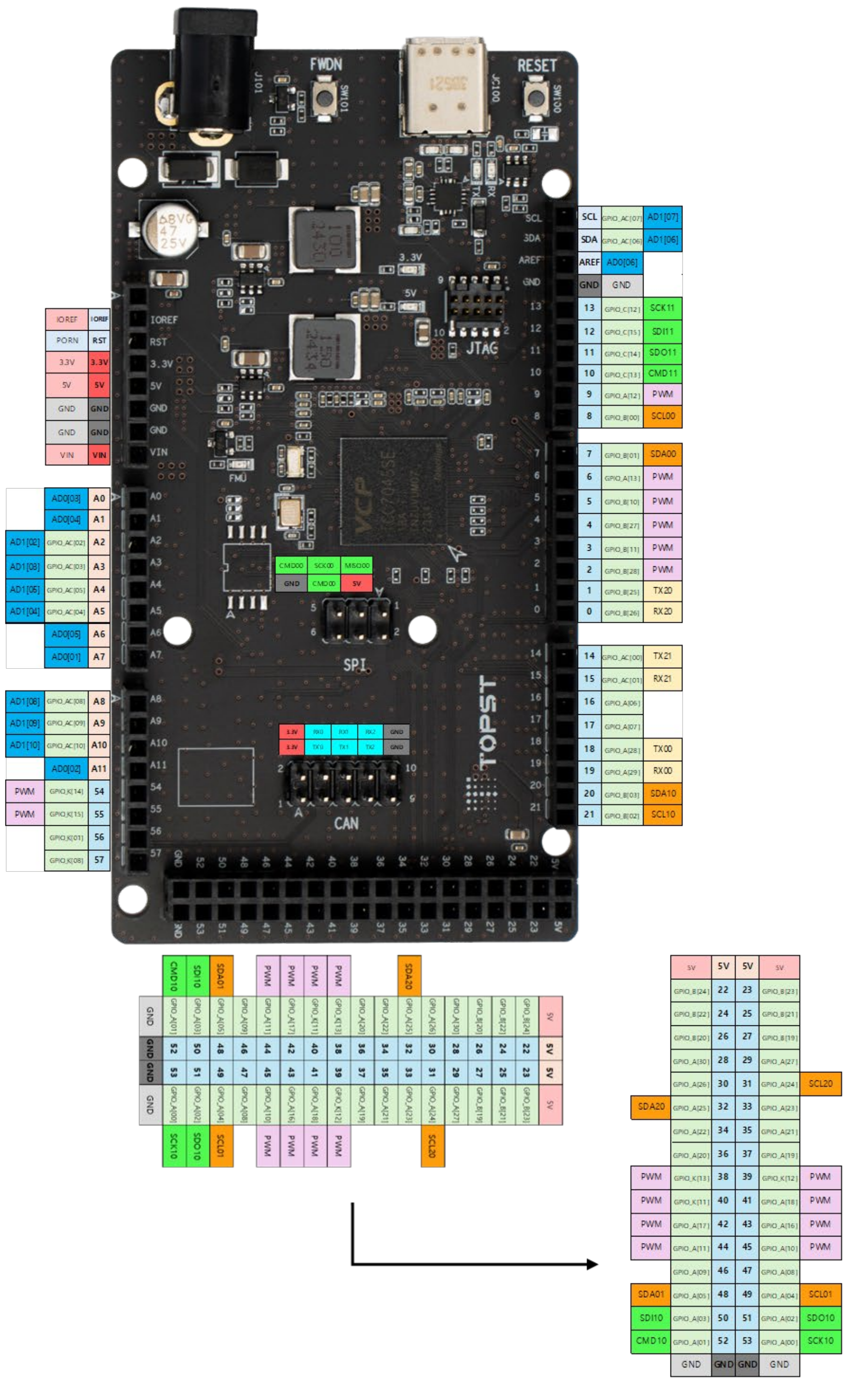

Figure 1.1 VCP-G Pinout Diagram

2 VCP-G Digital Pins

This chapter provides examples of controlling LEDs using the digital pins of the VCP-G board. On the VCP-G board, digital pins are used to send or receive binary signals (HIGH or LOW), making them essential for controlling components like LEDs, switches, and sensors.

This chapter includes two example projects that demonstrate how to use digital output to control multiple LEDs, providing a foundational understanding of digital pin functionality.

2.1 vcp4LED

This example program demonstrates how the VCP-G board controls four LEDs on the breadboard. The example code is provided in the "vcp4LED.ino" file. When this file is uploaded to the VCP-G board, the LEDs turn on and off sequentially in both forward and reverse patterns with a 500 ms delay between each transition.

2.1.1 Hardware Requirements

- VCP-G Board (x1)
- Breadboard (x1)
- LEDs (x4)
- 220Ω Resistors (x4)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x9)

2.1.2 Circuit

- LED01
  - (+) pin is connected to the 5V power rail on the breadboard with a 220Ω resistor, which is supplied by the VCP-G board.
  - (-) pin is connected to pin 47 on the VCP-G board.
- LED02
  - (+) pin is connected to the 5V power rail on the breadboard with a 220Ω resistor.
  - (-) pin is connected to pin 17 on the VCP-G board.
- LED03
  - (+) pin is connected to the 5V power rail on the breadboard with a 220Ω resistor.
  - (-) pin is connected to pin 50 on the VCP-G board.
- LED04
  - (+) pin is connected to the 5V power rail on the breadboard with a 220Ω resistor.
  - (-) pin is connected to pin 48 on the VCP-G board.

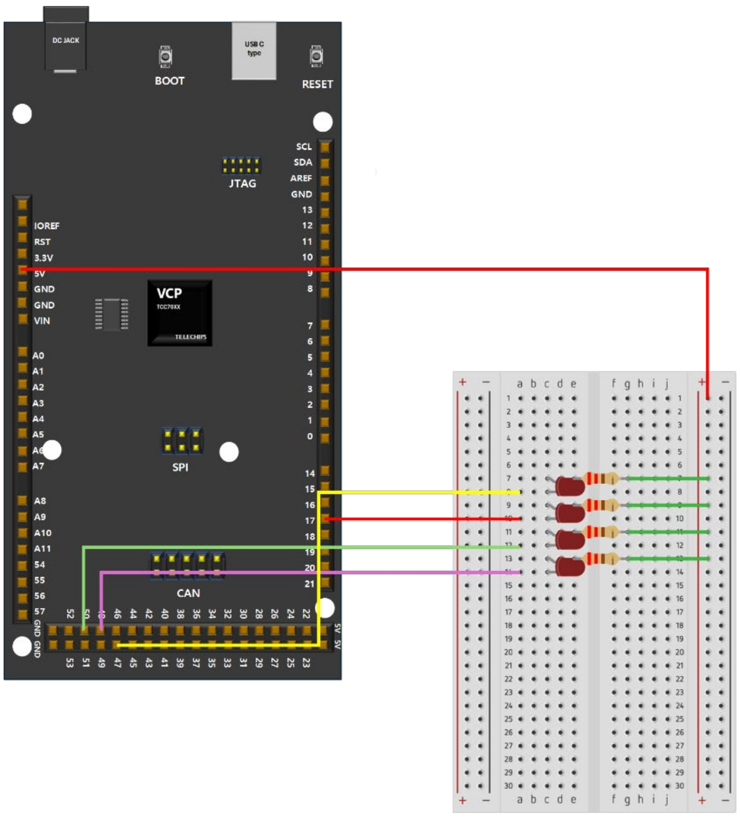

Figure 2.1 vcp4LED Circuit Schematic

2.1.2.1 Pin Mapping

Table 2.1 describes the pin mapping of vcp4LED.

Table 2.1 Pin Mapping of vcp4LED

Pin Name	VCP-G Board	Arduino IDE
LED01 (-) pin	47	47
LED02 (-) pin	17	17
LED03 (-) pin	50	50
LED04 (-) pin	48	48

2.1.3 How to execute

1. Open the "vcp4LED.ino" file.
   1) Open the Arduino IDE.
   2) Click **File → Examples → 01. VCP-G Digital → vcp4LED.**

```
/*
*  TOPST VCP : 4 LED Control
*/

int ledPin[] = {47, 17, 50, 48};

// the setup function runs once when you press reset or power the board
void setup() {
  // LED Mode and Init
  for(int i=0; i<4; i++)
  {
    pinMode(ledPin[i], OUTPUT);
    digitalWrite(ledPin[i], HIGH);
  }

}

// the loop function runs over and over again forever
void loop() {

  for(int i=0; i<4; i++)
  {
    digitalWrite(ledPin[i], LOW);
    delay(500);
  }

  for(int i=3; i>=0; i--)
  {
    digitalWrite(ledPin[i], HIGH);
    delay(500);
  }

}
```

2. Verify and upload the "vcp4LED.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
   If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include **`vcp4LED.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\C05357299384CE5734F0E696C5A4DA3B/vcp4LED.ino.rom

[main:155] Complete FWDN
```

2.2 vcp4LED_Button

This example program demonstrates how the VCP-G board controls four LEDs and a button on the breadboard. When the button is pressed, the right two LEDs turn off, and the left two LEDs turn on. When the button is released, the LEDs that were on turn off, and the LEDs that were off turn on. The program continuously checks the button state and adjusts the LEDs accordingly.

2.2.1 Hardware Requirements

- VCP-G Board (x1)
- Breadboard (x1)
- LEDs (x4)
- Button Switch (sensor) (x1)
- 220Ω Resistors (x4)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x12)

2.2.2 Circuit

- LED01
  - (+) pin is connected to the 5V power rail on the breadboard with a 220Ω resistor, which is supplied by the VCP-G board.
  - (-) pin is connected to pin 47 on the VCP-G board.
- LED02
  - (+) pin is connected to the 5V power rail on the breadboard with a 220Ω resistor.
  - (-) pin is connected to pin 17 on the VCP-G board.
- LED03
  - (+) pin is connected to the 5V power rail on the breadboard with a 220Ω resistor.
  - (-) pin is connected to pin 50 on the VCP-G board.
- LED04
  - (+) pin is connected to the 5V power rail on the breadboard with a 220Ω resistor.
  - (-) pin is connected to pin 48 on the VCP-G board.
- Button switch
  - One leg of the button switch is connected to pin 45 on the VCP-G board.
  - The diagonally opposite leg of the button is connected to the GND pin.

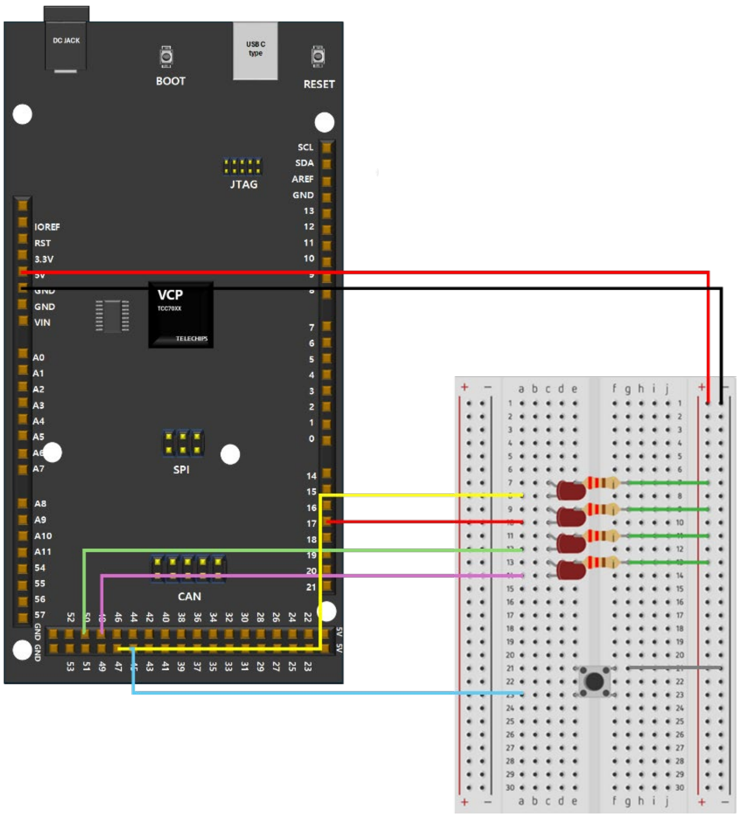

Figure 2.2 vcp4LED_Button Circuit Schematic

2.2.2.1 Pin Mapping

Table 2.2 describes the pin mapping of vcp4LED_Button.

Table 2.2 Pin Mapping of vcp4LED_Button

Pin Name	VCP-G Board	Arduino IDE
LED01 (-) pin	47	47
LED02 (-) pin	17	17
LED03 (-) pin	50	50
LED04 (-) pin	48	48
One leg pin of button	45	45

2.2.3 How to execute

1. Open the "vcp4LED_Button" file.
    1) Open the Arduino IDE.
    2) Click **File → Examples → 01. VCP-G Digital → vcp4LED_Button.**

```
/*
*  TOPST VCP : 4 LED and Button Control
*/

int ledPin[] = {47, 17, 50, 48};
int buttonPin[] = {45};

// the setup function runs once when you press reset or power the board
void setup() {
  // LED Mode and Init
  for(int i=0; i<4; i++)
  {
    pinMode(ledPin[i], OUTPUT);
    digitalWrite(ledPin[i], HIGH);
  }

  pinMode(buttonPin[0], INPUT);

}

// the loop function runs over and over again forever
void loop() {

  int button = digitalRead(buttonPin[0]);
  if(button==HIGH)
  {
    for(int i=0; i<2; i++)
    {
      digitalWrite(ledPin[i], LOW);
      digitalWrite(ledPin[i+2], HIGH);
    }
  }
  else
  {
    for(int i=3; i>=2; i--)
    {
      digitalWrite(ledPin[i-2], HIGH);
      digitalWrite(ledPin[i], LOW);
    }
  }

}
```

2. Verify and upload the "vcp4LED_Button.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
    1) Disconnect the power cable from the VCP-G board.
    2) Press and hold the FWDN button.
    3) Reconnect the power cable while continuing to hold the FWDN button.
    4) Release the FWDN button.
    If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include **`vcp4LED_Button.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\5CC1DB4CA216E2BC009504FAA3D06456/vcp4LED_Button.ino.rom

[main:155] Complete FWDN
```

3 VCP-G ANALOG

This chapter provides examples of using the analog pins on the VCP-G board. On the VCP-G board, analog pins receive continuous voltage signals from sensors, allowing precise measurement of varying input values. Chapter 3.1, Chapter 3.2, and Chapter 3.3 describe how to use analog pins to read sensor data and control outputs, providing a foundational understanding of analog input handling.

3.1 AnalogInOutSerial

This example program demonstrates how the VCP-G board controls a potentiometer and an LED on the breadboard. The VCP-G board reads a value from an analog input pin, maps the result to a range from 0 to 1000, and uses this value to set the pulse width modulation (PWM) of an output pin (connected to an LED). The results are also printed to the Serial Monitor.

3.1.1 Hardware Requirements

- VCP-G Board (x1)
- Breadboard (x1)
- LED (x1)
- Potentiometer (x1)
- 220Ω Resistors (x2)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x4)

3.1.2 Circuit

- Potentiometer
  - Center pin of the potentiometer is connected to the analog pin A5 on the VCP-G board.
  - GND pin of the potentiometer is connected to pin 43 on the VCP-G board and is connected to GND pin on the VCP-G board with a 220Ω resistor.
- LED
  - (+) pin of the LED is connected to 3.3V on the VCP-G board with a 220Ω resistor.
  - (-) pin of the LED is connected to center pin of the potentiometer.

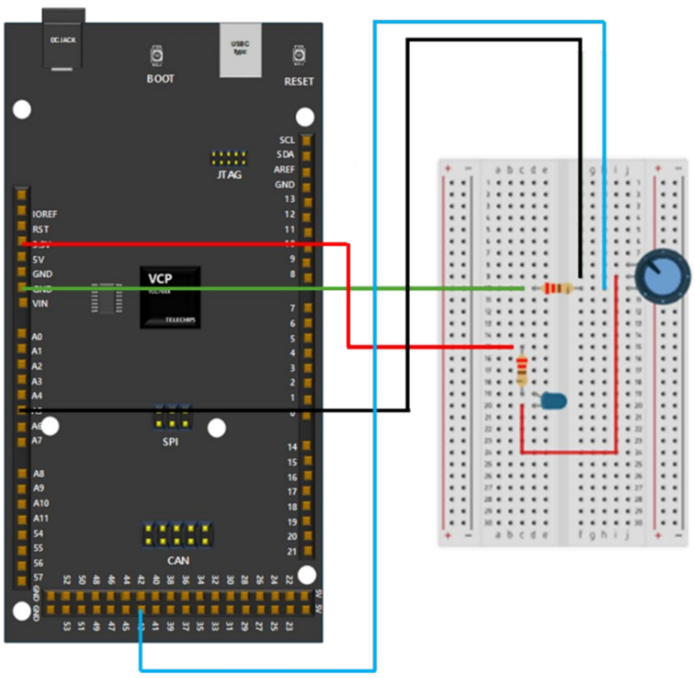

Figure 3.1 AnalogInOutSerial Circuit Schematic

3.1.2.1 Pin Mapping

Table 3.1 describes the pin mapping of AnalogInOutSerial.

Table 3.1 Pin Mapping of AnalogInOutSerial

Pin Name	VCP-G Board	Arduino IDE
Potentiometer Center pin	A5	-
Potentiometer GND pin	43	43
LED (+) pin	3.3V	-
LED (-) pin	A5	A5

3.1.3 How to execute

1. Open the "AnalogInOutSerial.ino" file.
   1) Open the Arduino IDE.
   2) Click **File → Examples → 02. VCP-G Analog → AnalogInOutSerial.**

```
#include <HardwareSerial.h>
HardwareSerial Serial;

// These constants won't change. They're used to give names to the pins used:
const int analogInPin = 43;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = A5;  // Analog output pin that the LED is attached to

int sensorValue = 0;  // value read from the pot
int outputValue = 0;  // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 115200 bps:
  Serial.begin(115200);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 4095, 0, 1000);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue / 4);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(2);
}
```

2. Verify and upload the "AnalogInOutSerial.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include **`AnalogInOutSerial.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\EB016432EF98DEF0B9102FD77148DD5D/AnalogInOutSerial.ino.
```

```
rom

[main:155] Complete FWDN
```

3.2 AnalogInput

This example program demonstrates how the VCP-G board controls a potentiometer and an LED on the breadboard. The VCP-G board reads a value from an analog input pin and uses this value to control an LED. If the sensor value is less than 3000, the LED turns on. If the sensor value is 3000 or higher, the LED turns off.

3.2.1 Hardware Requirements

- VCP-G Board (x1)
- Breadboard (x1)
- LED (x1)
- Potentiometer (x1)
- 220Ω Resistor (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x6)

3.2.2 Circuit

- Potentiometer
  - VCC pin of the potentiometer is connected to 3.3V on the VCP-G board with a 220Ω resistor.
  - Center pin of the potentiometer is connected to the analog pin A5 on the VCP-G board.
  - GND pin of the potentiometer is connected to GND pin on the VCP-G board with a 220Ω resistor.
- LED
  - (+) pin of the LED is connected to 3.3V on the VCP-G board with a 220Ω resistor.
  - (-) pin of the LED is connected to pin 5 on the VCP-G board.

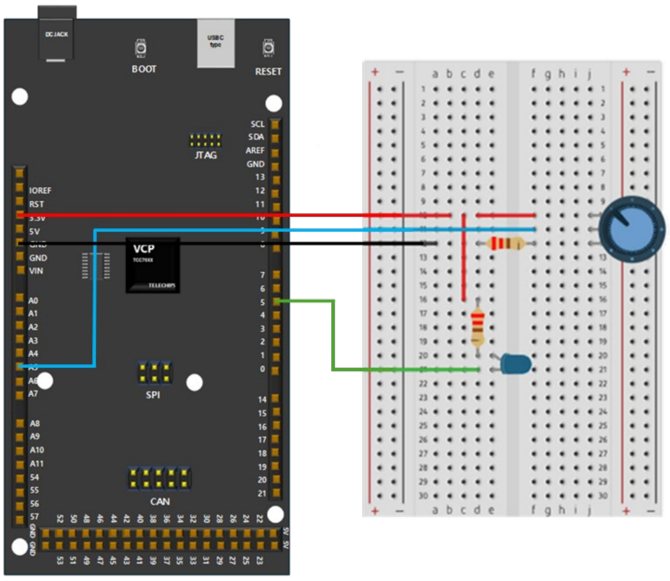

Figure 3.2 AnalogInput Circuit Schematic

3.2.2.1 Pin Mapping

Table 3.2 describes the pin mapping of AnalogInput.

Table 3.2 Pin Mapping of AnalogInput

Pin Name	VCP-G Board	Arduino IDE
Potentiometer VCC pin	3.3V	-
Potentiometer Center pin	A5	A5
Potentiometer GND pin	GND	-
LED (+) pin	3.3V	-
LED (-) pin	5	5

3.2.3 How to execute

1. Open the "AnalogInput.ino" file.
   1) Open the Arduino IDE.
   2) Click **File → Examples → 02. VCP-G Analog → AnalogInput.**

```
#include <HardwareSerial.h>
HardwareSerial Serial;

int sensorPin = A5;
int ledPin = 5;
int sensorValue = 0;

void setup()
{
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  for(;;)
  {
    sensorValue = analogRead(sensorPin);
    Serial.println(sensorValue);
    if(sensorValue<3000)
    {
      digitalWrite(ledPin, HIGH);
    }
    else
    {
      digitalWrite(ledPin, LOW);
    }
    delay(10);
  }
}
```

Note 1: To check **`sensorValue`** in the Serial Monitor, add **`Serial.println()`** to the source code.
Note 2: A fixed resistor is used along with a variable resistor (potentiometer) to adjust the sensor value. The sensor value changes depending on how much the potentiometer is turned, and the amount you need to turn the potentiometer varies based on the value of the fixed resistor.

2. Verify and upload "AnalogInput.ino" file to VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
   If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include **`AnalogInput.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\C3FDEE51354320EA689DFEB4EDCF2ECD/AnalogInput.ino.rom

[main:155] Complete FWDN
```

3.3 pwmFade

This example program demonstrates how the VCP-G board controls an LED on the breadboard by gradually increasing and decreasing its brightness in a loop using PWM. After the LED reaches its maximum brightness, the brightness of the LED begins to decrease. The program continuously adjusts the LED's brightness, creating a fading effect.

3.3.1 Hardware Requirements

- VCP-G Board (x1)
- Breadboard (x1)
- LED (x1)
- 220Ω Resistor (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x2)

3.3.2 Circuit

- LED
  - (+) pin of the LED is connected to 5V on the VCP-G board.
  - (-) pin of the LED is connected to pin 9 on the VCP-G board with a 220Ω resistor.

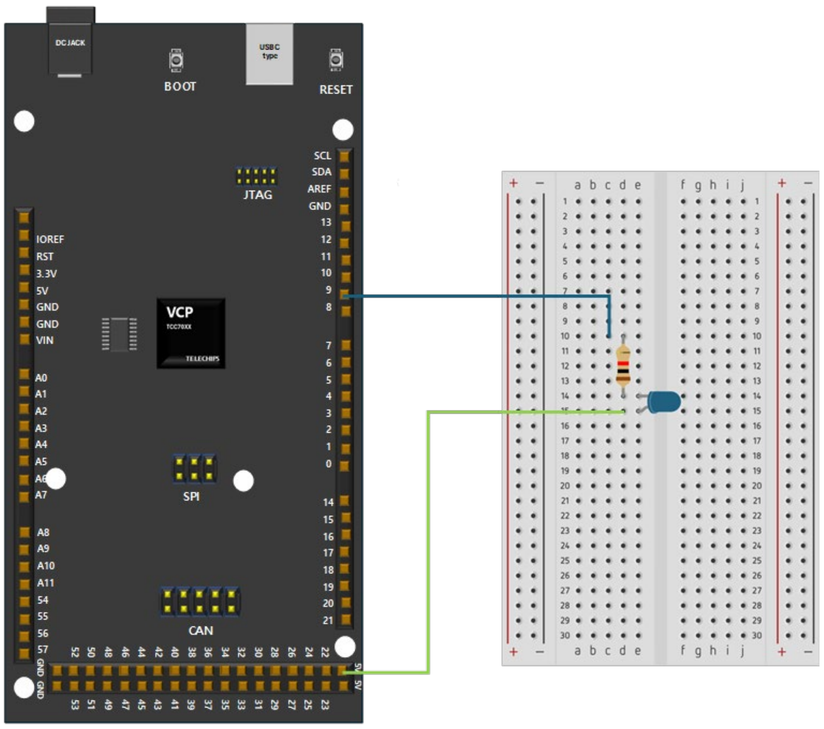

Figure 3.3 pwmFade Circuit Schematic

3.3.2.1 Pin Mapping

Table 3.3 describes the pin mapping of pwmFade.

Table 3.3 Pin Mapping of pwmFade

Pin Name	VCP-G Board	Arduino IDE
LED (+) pin	5V	-
LED (-) pin	9	9

3.3.3 How to execute

1. Open the "pwmFade.ino" file.
   1) Open the Arduino IDE.
   2) Click **File → Examples → 02. VCP-G Analog → pwmFade.**

```
/*
  Fade

  This example shows how to fade an LED on pin 11 using the analogWrite()
  function.

  The analogWrite() function uses PWM, so if you want to change the pin you're
  using, be sure to use another PWM capable pin.
*/

int led = 9;         // the PWM pin the LED is attached to
int brightness = 0;  // how bright the LED is
int fadeAmount = 5;  // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

2. Verify and upload the "pwmFade.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include `pwmFade.ino.rom`.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\69446E8A7F6616A7D5466014BDF759FC/pwmFade.ino.rom

[main:155] Complete FWDN
```

4 VCP-G SPI

This chapter provides instructions for configuring Serial Peripheral Interface (SPI) communication on the VCP-G board.
SPI is a high-speed, synchronous communication protocol used to exchange data between microcontrollers and peripherals. It operates with separate lines for data transmission (MOSI and MISO), clock synchronization (SCK), and device selection (SS), ensuring efficient and reliable communication.

The following chapters describe how to set up and use SPI to interface with external devices.

4.1 vcpSPI_Dot8x8

This example program demonstrates how the VCP-G board controls an 8x8 LED dot matrix using the MAX7219 driver. The 8x8 LED dot matrix displays patterns such as a heart shape and the letter "R" by setting rows with pre-defined binary arrays. The intensity of the LEDs is adjusted to create a pulsing effect, adding dynamic visuals. Additional features include inverting and clearing the display to enhance functionality.

4.1.1 Hardware Requirements

- VCP-G Board (x1)
- 8x8 Dot Matrix (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-socket Jumper Wires (x5)

4.1.2 Circuit

- 8x8 Dot Matrix
  - VCC pin of the 8x8 Dot Matrix is connected to the analog pin 5V on the VCP-G board.
  - GND pin of the 8x8 Dot Matrix is connected to GND on the VCP-G board.
  - DIN pin of the 8x8 Dot Matrix is connected to pin 11 on the VCP-G board.
  - CS pin of the 8x8 Dot Matrix is connected to pin 10 on the VCP-G board.
  - CLS pin of the 8x8 Dot Matrix is connected to pin 13 on the VCP-G board.

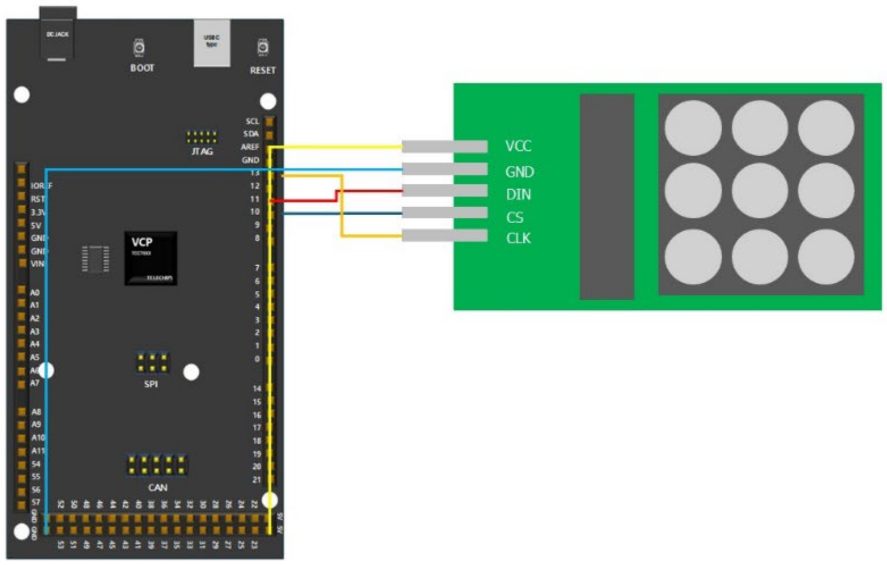

Figure 4.1 vcpSPI_Dot8x8 Circuit Schematic

4.1.2.1 Pin Mapping

Table 4.1 describes the pin mapping of vcpSPI_Dot8x8.

Table 4.1 Pin Mapping of vcpSPI_Dot8x8

Pin Name	VCP-G Board	Arduino IDE
8x8 Dot Matrix VCC pin	5V	-
8x8 Dot Matrix GND pin	GND	-
8x8 Dot Matrix DIN pin	11	11
8x8 Dot Matrix CS pin	10	10
8x8 Dot Matrix CLK pin	13	13

4.1.3 How to execute

1. Open the "vcpSPI_Dot8x8.ino" file.
   1) Open the Arduino IDE.
   2) Click **File → Examples → 03. VCP-G SPI → vcpSPI_Dot8x8.**

```
#include <MAX7219.h>

// 8x8 Dot-Matrix
#define DATAPIN     11 // SPI1_DO
#define CLOCKPIN    13 // SPI1_CLK
#define LOADPIN     10 // SPI1_CS

// binary representation of a heart shape
const static byte HEART[8] = {
  0b01100110,
  0b10011001,
  0b10000001,
  0b10000001,
  0b01000010,
  0b00100100,
  0b00011000,
  0b00000000
};

// The Letter "R"
// human-readable binary representation from left-to-right and top-to-bottom
const static byte R[8] = {
  0b00000000,
  0b01111100,
  0b01000110,
  0b01111100,
  0b01001000,
  0b01000100,
  0b01000010,
  0b00000000
};

void setup() {

}

void loop() {
  MAX7219 Matrix(1, DATAPIN, CLOCKPIN, LOADPIN);
  // First LED in upper left Corner
  Matrix.setLed(1, 0, 0, true);
  delay(500);
  Matrix.setLed(1, 0, 0, false);
  delay(500);

  // Make Heartshape from array
  for(int row = 0; row <= 7; row++)
  {
    Matrix.setRow(1, row, HEART[row]);
    delay(100);
  }
  delay(500);

  // Change Intensity to make it pulse
  for(int repeats = 0; repeats < 3; repeats++)
  {
    // increase intensity
    for(int i = 1; i <= 15; i++)
    {
      Matrix.setIntensity(1, i);
      delay(20);
    }

    // decrease intensity
    for(int j = 15; j >= 1; j--)
    {
      Matrix.setIntensity(1, j);
      delay(20);
    }
  }
  delay(500);

  // Make Letter "R" from array
  for(int row = 0; row <= 7; row++)
  {
    Matrix.setRow(1, row, R[row]);
    delay(100);
  }
  delay(1000);

  // invert display
  Matrix.invertDisplay(1);
  delay(1000);

  // clear display
  Matrix.clearDisplay(1);
  delay(1000);
}
```

2. Verify and upload the "vcpSPI_Dot8x8.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include **`vcpSPI_Dot8x8.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\567805554B5B9F915DCC80B38483AE07/vcpSPI_Dot8x8.ino.rom

[main:155] Complete FWDN
```

Note: If you want to use the SPI pin in the center of the VCP-G board, you can use it by referring to the pin number as shown in Table 4.2.

Table 4.2 Mapping of Center SPI Pin on VCP-G Board

Pin Name	VCP-G Board	Arduino IDE
1	MISO	58
2	5V	-
3	SCK	59
4	MOSI	60

Pin Name	VCP-G Board	Arduino IDE
5	CMD	61
6	GND	-

5 VCP-G I2C

This chapter provides instructions for configuring Inter-integrated Circuit (I2C) communication on the VCP-G board.
I2C is a two-wire, synchronous communication protocol designed for efficient data exchange between multiple devices. It operates with a serial data line (SDA) and a serial clock line (SCL), allowing multiple peripherals to communicate with a microcontroller by using unique addresses. I2C supports both master-slave communication and multi-master configurations, making it ideal for connecting sensors, displays, and other low-speed devices while minimizing the number of required connections.

5.1 vcpI2C_LCD1602

This example program demonstrates how the VCP-G board controls an LCD1602 display using the I2C communication protocol. The LCD1602 is a 16-character, 2-line liquid crystal display commonly used in embedded system projects. By utilizing the LiquidCrystal_I2C library, the board sends commands and data over the I2C bus to efficiently control the display.

In this example, the LCD is initialized, and the backlight is enabled for clear visibility. The program then positions the cursor to display the text **`"VCP-G"`** on the first row and **`"I2C Test!"`** on the second row. With I2C communication, multiple devices can be controlled using minimal wiring, making it an effective solution for compact projects.

5.1.1 Hardware Requirements

- VCP-G Board (x1)
- LCD1602 (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-socket Jumper Wires (x4)

5.1.2 Circuit

- LCD1602
  - VCC pin of the LCD1602 is connected to the analog pin 5V on the VCP-G board.
  - GND pin of the LCD1602 is connected to GND on the VCP-G board.
  - SDA pin of the LCD1602 is connected to pin 48 on the VCP-G board.
  - SCL pin of the LCD1602 is connected to pin 49 on the VCP-G board.

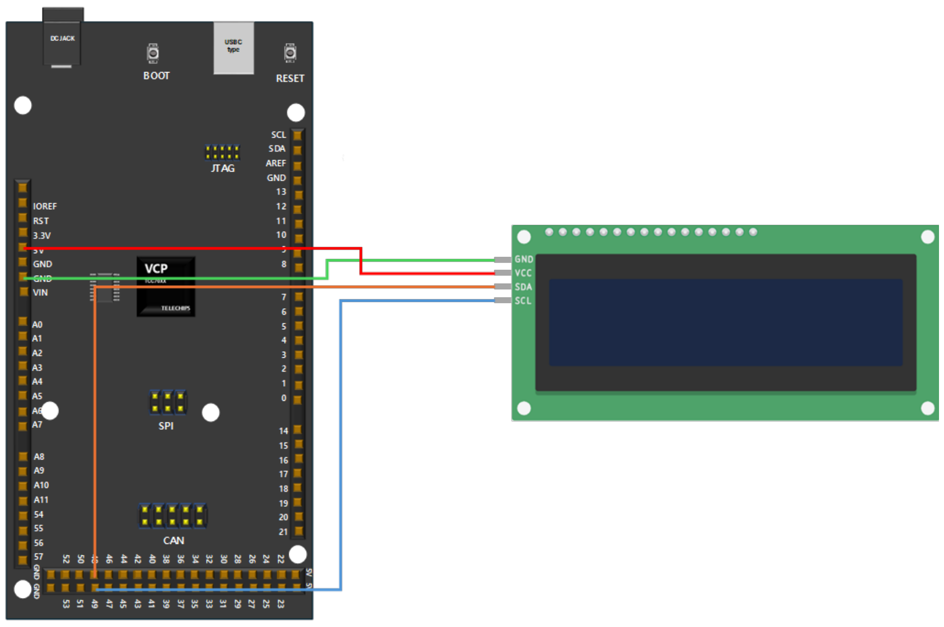

Figure 5.1 vcpI2C_LCD1602 Circuit Schematic

5.1.2.1 Pin Mapping

Table 5.1 describes the pin mapping of vcpI2C_LCD1602.

Table 5.1 Pin Mapping of vcpI2C_LCD1602

Pin Name	VCP-G Board	Arduino IDE
LCD1602 VCC pin	5V	-
LCD1602 GND pin	GND	-
LCD1602 SDA pin	48	-
LCD1602 SCL pin	49	-

5.1.3 How to execute

1. Open the "vcpI2C_LCD1602.ino" file.
   1) Open the Arduino IDE.
   2) Click **File → Examples → 04. VCP-G I2C → vcpI2C_LCD1602.**

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line display

void setup()
{
  LiquidCrystal_I2C lcd(0x27,16,2);

  lcd.init();
  lcd.backlight(); // initialize the lcd

  lcd.setCursor(2, 0);
  delay(10);
  lcd.print("* TOPST VCP-G *");

  lcd.setCursor(4, 1);
  delay(10);
  lcd.print("I2C Test!");
}

void loop()
{

}
```

2. Verify and upload the "vcpI2C_LCD1602.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
   If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include `vcpI2C_LCD1602.ino.rom`.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\C8D91A6857B651D6C665B0EF18B7EE53/vcpI2C_LCD1602.ino.rom

[main:155] Complete FWDN
```

6 VCP-G UART

This chapter provides instructions for configuring Universal Asynchronous Receiver-Transmitter (UART) communication on the VCP-G board.

UART is a widely used serial communication protocol that transmits data asynchronously using only two lines: Transmit (TX) and Receive (RX). It is essential for exchanging data between microcontrollers, sensors, and computers without requiring a shared clock signal.

The following chapters describe how to send and receive data through UART.

6.1 vcpASCIITable

This example program demonstrates how the VCP-G board prints the ASCII values of characters in various formats: decimal, hexadecimal, octal, and binary. It starts from the character '!' (ASCII value 33) and increments through all visible ASCII characters, printing each in different formats. The program continues until it reaches the character '~' (ASCII value 126).

6.1.1 Hardware Requirements

- VCP-G Board (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)

6.1.2 How to execute

1. Open the "vcpASCIITable.ino" file.
   1) Open the Arduino IDE.
   2) Click **File → Examples → 05. VCP-G UART → vcpASCIITable.**

```
/*
  ASCII table

  Prints out byte values in all possible formats:
  - as raw binary values
  - as ASCII-encoded decimal, hex, octal, and binary values

  For more on ASCII, see http://www.asciitable.com and http://en.wikipedia.org/wiki/ASCII

  The circuit: No external hardware needed.

  created 2006
  by Nicholas Zambetti <http://www.zambetti.com>
  modified 9 Apr 2012
  by Tom Igoe

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/ASCIITable
*/

#include "HardwareSerial.h"
HardwareSerial Serial;

void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(115200);
  while (!Serial) {
    ;  // wait for serial port to connect. Needed for native USB port only
  }

  // prints title with ending line break
  Serial.println("ASCII Table ~ Character Map");
```

```
}

// first visible ASCIIcharacter '!' is number 33:
int thisByte = 33;
// you can also write ASCII characters in single quotes.
// for example, '!' is the same as 33, so you could also use this:
// int thisByte = '!';

void loop() {
  // prints value unaltered, i.e. the raw binary version of the byte.
  // The Serial Monitor interprets all bytes as ASCII, so 33, the first number,
  // will show up as '!'

  Serial.print(", dec: ");
  // prints value as string as an ASCII-encoded decimal (base 10).
  // Decimal is the default format for Serial.print() and Serial.println(),
  // so no modifier is needed:
  Serial.print(thisByte);
  // But you can declare the modifier for decimal if you want to.
  // this also works if you uncomment it:

  // Serial.print(thisByte, DEC);

  Serial.print(", hex: ");
  // prints value as string in hexadecimal (base 16):
  Serial.print(thisByte, HEX);

  Serial.print(", oct: ");
  // prints value as string in octal (base 8);
  Serial.print(thisByte, OCT);

  Serial.print(", bin: ");
  // prints value as string in binary (base 2) also prints ending line break:
  Serial.println(thisByte, BIN);

  // if printed last visible character '~' or 126, stop:
  if (thisByte == 126) {  // you could also use if (thisByte == '~') {
    // This loop loops forever and does nothing
    while (true) {
      continue;
    }
  }
  // go on to the next character
  thisByte++;
}
```

2. Verify and upload "vcpASCIITable.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include `vcpASCIITable.ino.rom`.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topstAppData\Local\arduino\sketches\487F45098412336AA9D73C50C17E07D8/vcpASCIITable.ino.rom

[main:155] Complete FWDN
```

6.2 vcpGraph

This example program demonstrates how the VCP-G board reads analog value of potentiometer on the breadboard and transmits the data to host PC through UART. The Arduino code continuously reads the value of the analog sensor (potentiometer) connected to pin A5 and sends it through the serial port. The accompanying processing code visualizes these values in a dynamic graph in real time, showing the change in sensor input over time.

6.2.1 Hardware Requirements

- VCP-G Board (x1)
- Breadboard (x1)
- Potentiometer (x1)
- 10 kΩ Resistor (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x4)

6.2.2 Circuit

- Potentiometer
  - Center pin of the potentiometer is connected to the analog pin A5 on the VCP-G board.
  - GND pin of the potentiometer is connected to GND on the VCP-G board with a 10 kΩ resistor.
  - VCC pin of the potentiometer is connected to 3.3V on the VCP-G board.

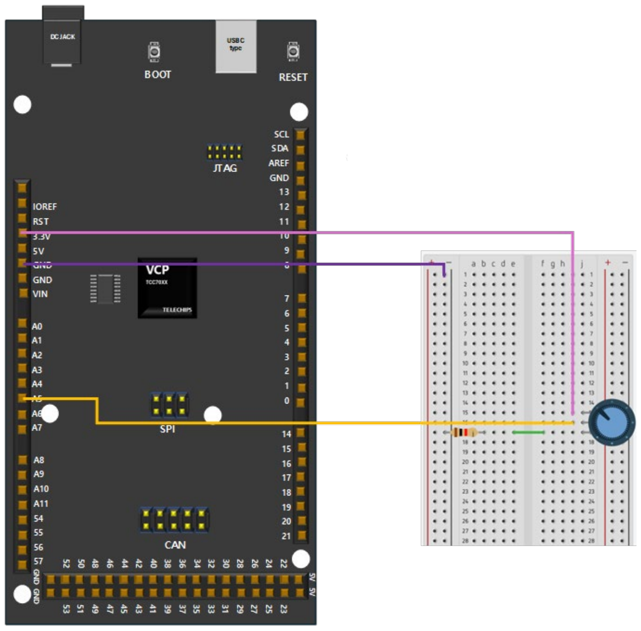

Figure 6.1 vcpGraph Circuit Schematic

6.2.2.1 Pin Mapping

Table 6.1 describes the pin mapping of vcpGraph.

Table 6.1 Pin Mapping of vcpGraph

Pin Name	VCP-G Board	Arduino IDE
Potentiometer Center pin	A5	A5
Potentiometer GND pin	GND	-
Potentiometer VCC pin	3.3V	-

6.2.3 How to execute

1. Open the "vcpGraph.ino" file.
   1) Open the Arduino IDE.
   2) Click **File → Examples → 05 VCP-G UART → vcpGraph.**

```
/*
  Graph

  A simple example of communication from the Arduino board to the computer: The
  value of analog input 0 is sent out the serial port. We call this "serial"
  communication because the connection appears to both the Arduino and the
  computer as a serial port, even though it may actually use a USB cable. Bytes
  are sent one after another (serially) from the Arduino to the computer.

  You can use the Arduino Serial Monitor to view the sent data, or it can be
  read by Processing, PD, Max/MSP, or any other program capable of reading data
  from a serial port. The Processing code below graphs the data received so you
  can see the value of the analog input changing over time.

  The circuit:
  - any analog input sensor attached to analog in pin 0

  created 2006
  by David A. Mellis
  modified 9 Apr 2012
  by Tom Igoe and Scott Fitzgerald

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Graph
*/

#include "HardwareSerial.h"
HardwareSerial Serial;

void setup() {
  // initialize the serial communication:
  Serial.begin(115200);
  for(;;)
  {
    Serial.println(analogRead(A5));
    // wait a bit for the analog-to-digital converter to stabilize after the last
    // reading:
    delay(2);
  }
}

void loop() {
  // send the value of analog input 0:
  // Serial.println(analogRead(A0));
  // // wait a bit for the analog-to-digital converter to stabilize after the last
  // // reading:
  // delay(2);
}

/* Processing code for this example

  // Graphing sketch

  // This program takes ASCII-encoded strings from the serial port at 9600 baud
  // and graphs them. It expects values in the range 0 to 1023, followed by a
  // newline, or newline and carriage return

  // created 20 Apr 2005
  // updated 24 Nov 2015
  // by Tom Igoe
  // This example code is in the public domain.
  import processing.serial.*;

  Serial myPort;        // The serial port
  int xPos = 1;         // horizontal position of the graph
  float inByte = 0;

  void setup () {
    // set the window size:
    size(400, 300);

    // List all the available serial ports
    // if using Processing 2.1 or later, use Serial.printArray()
    println(Serial.list());

    // I know that the first port in the serial list on my Mac is always my
    // Arduino, so I open Serial.list()[0].
    // Open whatever port is the one you're using.
    myPort = new Serial(this, Serial.list()[0], 9600);

    // don't generate a serialEvent() unless you get a newline character:
    myPort.bufferUntil('\n');

    // set initial background:
    background(0);
  }

  void draw () {
    // draw the line:
    stroke(127, 34, 255);
    line(xPos, height, xPos, height - inByte);

    // at the edge of the screen, go back to the beginning:
    if (xPos >= width) {
      xPos = 0;
      background(0);
    } else {
      // increment the horizontal position:
      xPos++;
    }
  }

  void serialEvent (Serial myPort) {
    // get the ASCII string:
    String inString = myPort.readStringUntil('\n');

    if (inString != null) {
      // trim off any whitespace:
      inString = trim(inString);
      // convert to an int and map to the screen height:
      inByte = float(inString);
      println(inByte);
      inByte = map(inByte, 0, 1023, 0, height);
    }
  }

*/
```

2. Verify and upload the "vcpGraph.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After a successful upload, check the Arduino IDE output console for the following message:
   Note: The message should include **`vcpGraph.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\F59E4532EC3A529F5910F376F809A5E5/vcpGraph.ino.rom

[main:155] Complete FWDN
```

7 Additional Examples

This chapter provides additional sensor examples not included in the "Examples for TOPST VCP Rev G" in the Arduino IDE.
It provides example guides for using commonly used Arduino sensors with the VCP-G board, enabling you to integrate various sensors into your projects effectively.

7.1 Infrared (IR) Sensor (Transceiver)

7.1.1 Infrared (IR) Sensor 1

This example demonstrates how the VCP-G board controls an IR sensor and two LEDs on the breadboard. After reading the IR sensor value, if the IR sensor value is HIGH, it is considered that there is no obstacle, and the green LED turns on while the red LED turns off. Conversely, if the IR sensor value is LOW, it is considered that there is an obstacle, and the red LED turns on while the green LED turns off. Additionally, the presence or absence of an obstacle is displayed on the serial monitor.

7.1.1.1 Hardware Requirements

- VCP-G Board (x1)
- Breadboard (x1)
- IR Transceiver Sensor (x1)
- LEDs x2 (Different colors are recommended)
- 220Ω Resistors (x2)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x4)
- Pin-to-socket Jumper Wires (x3)

7.1.1.2 Circuit

- IR Transceiver Sensor
  - OUT pin of the IR sensor is connected to pin 50 on the VCP-G board.
  - VCC pin of the IR sensor is connected to 5V on the VCP-G board.
  - GND pin of the IR sensor is connected to GND on the VCP-G board.
- Red LED
  - (-) of LED is connected to the resistor, and the resistor is connected to GND on the VCP-G board.
  - (+) of LED is connected to pin 48 on the VCP-G board.
- Green LED
  - (-) of LED is connected to the resistor, and the resistor is connected to GND on the VCP-G board.
  - (+) of LED is connected to pin 17 on the VCP-G board.

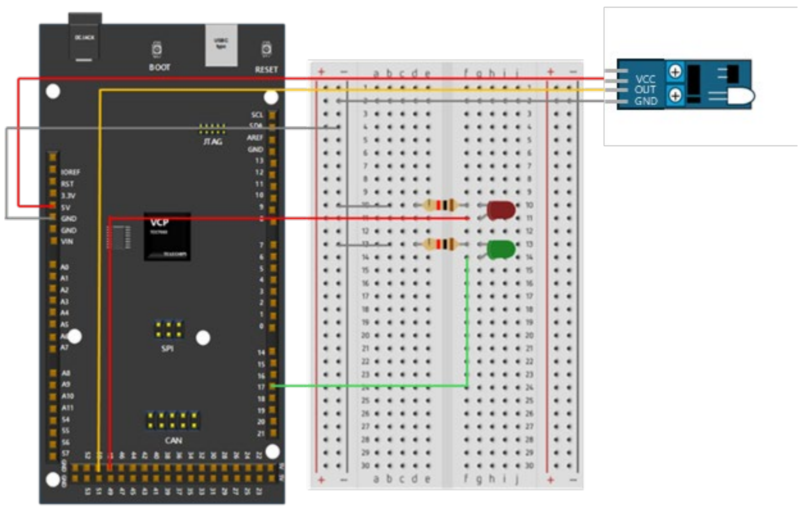

Figure 7.1 Infrared (IR) Sensor Circuit Schematic

7.1.1.2.1 Pin Mapping

Table 7.1 describes the pin mapping of irSensor_LED.

Table 7.1 Pin Mapping of irSensor_LED

Pin Name	VCP-G Board	Arduino IDE
IR Sensor OUT pin	50	50
IR Sensor VCC pin	5V	-
IR Sensor GND pin	GND	-
Red LED (+) pin	48	48
Red LED (-) pin	GND	-
Green LED (+) pin	17	17
Green LED (-) pin	GND	-

7.1.1.3 How to execute

1. Copy the following source code into the Arduino IDE and save the file as "irSensor_LED.ino".
   Note: The following source code is provided only in this document.

```
#include "HardwareSerial.h"
HardwareSerial Serial;

// Control LEDs based on obstacle detection using an IR sensor
int outputPin = 50, red = 48, green = 17;

void setup() {
  pinMode(outputPin, INPUT);
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  Serial.begin(115200);
}

void loop() {
  // Read the value from the IR sensor
  int value = digitalRead(outputPin);

  // Control LEDs based on the presence of an obstacle
  if (value == HIGH) {
    // No obstacle detected
    digitalWrite(red, LOW);
    digitalWrite(green, HIGH);
    Serial.println("No object detected.");
  }
  else {
    // Obstacle detected
    digitalWrite(red, HIGH);
    digitalWrite(green, LOW);
    Serial.println("Object detected!");
  }
}
```

2. Verify and upload the "irSensor_LED.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino output console for the following message:
   Note: The message should include **`irSensor_LED.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\1D003A72AB3391D6D60FD7993380F8CD/irSensor_LED.ino.rom
```

```
[main:155] Complete FWDN
```

7.1.2 Infrared (IR) Sensor 2

This example demonstrates how the VCP-G board controls an IR sensor to detect objects and print the detection status to the Serial Monitor. The IR Transceiver reads the presence of an obstacle. If the IR Transceiver value is HIGH, it indicates no obstacle, and the green LED turns on and the red LED turns off. Conversely, if the IR Transceiver value is LOW, it indicates there is an obstacle, and the red LED turns on and the green LED turns off. Additionally, the presence or absence of an obstacle is displayed on the serial monitor.

7.1.2.1 Hardware Requirements

- VCP-G Board (x1)
- IR Transceiver Sensor (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-socket Jumper Wires (x3)

7.1.2.2 Circuit

- IR Transceiver Sensor
  - OUT pin of the IR Transceiver sensor is connected to pin 8 on the VCP-G board.
  - VCC pin of the IR Transceiver sensor is connected to 5V on the VCP-G board.
  - GND pin of the IR Transceiver sensor is connected to GND to the VCP-G board.

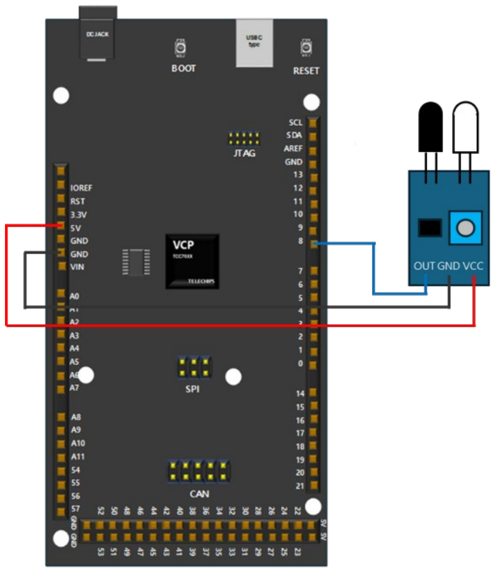

Figure 7.2 Infrared (IR) Sensor (Transceiver) Circuit Schematic

7.1.2.2.1 Pin Mapping

Table 7.2 describes the pin mapping of irTransceiver.

Table 7.2 Pin Mapping of irTransceiver

Pin Name	VCP-G Board	Arduino IDE
IR Transceiver Sensor OUT pin	8	8
IR Transceiver Sensor VCC pin	5V	-
IR Transceiver Sensor GND pin	GND	-

7.1.2.3 How to execute

1. Copy the following source code into the Arduino IDE and save the file as "irTransceiver.ino".
   Note: The following source code is provided only in this document.

```
/*
  This code prints "Detected" to the Serial Monitor when an object is detected,
  and "Not Detected" when no object is detected.
*/

#include "HardwareSerial.h"
HardwareSerial Serial;

int sensorValue = 8;
int val = 0;

void setup() {
  Serial.begin(9600);
  pinMode(sensorValue, INPUT);
}

void loop() {
  val = digitalRead(sensorValue);
  if (val == HIGH) {
    Serial.println("Not Detected");
  }
  else {
    Serial.println("Detected");
  }
}
```

2. Verify and upload the "irTransceiver.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include **`irTransceiver.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\1D003A72AB3391D6D60FD7993380F8CD/irTransceiver.ino.rom

[main:155] Complete FWDN
```

7.2 Joystick

This example shows how the VCP-G board reads joystick inputs and displays their value on the Serial Monitor. You can receive three inputs: X-axis, Y-axis, and buttons. The serial monitor verifies the received signals. The movements made on the X and Y axes change the value of the port, which corresponds to the numeric value of the analog output. It allows for precise control for applications that require fine adjustment.

Note: Dual Axis Joystick Module (KY-023) is a product of Joy-IT. All rights to its design, trademark, and related intellectual property are owned by Joy-IT.

7.2.1 Hardware Requirements

- VCP-G Board (x1)
- Dual Axis Joystick Module (KY-023) (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-socket Jumper Wires (x5)

7.2.2 Circuit

- KY-023 (Dual Axis Joystick Module)
  - 5V pin of KY-023 is connected to the 5V on the VCP-G board.
  - GND pin of KY-023 is connected to GND on the VCP-G board.
  - VRx pin of KY-023 is connected to the analog pin A5 on the VCP-G board.
  - VRy pin of KY-023 is connected to the analog pin A4 on the VCP-G board.
  - SW pin of KY-023 is connected to pin 2 on the VCP-G board.

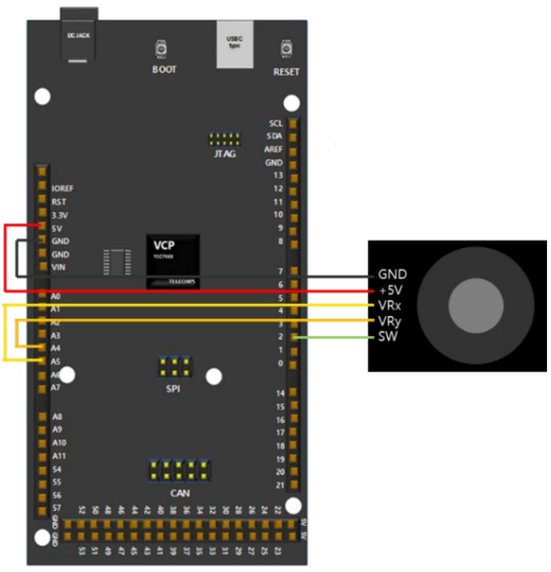

Figure 7.3 Joystick Circuit Schematic

7.2.2.1 Pin Mapping

Table 7.3 describes the pin mapping of Joystick.

Table 7.3 Pin Mapping of Joystick

Pin Name	VCP-G Board	Arduino IDE
VRx	A5	A5
VRy	A4	A4
SW	2	2
5V	5V	-
GND	GND	-

7.2.3 How to execute

1. Copy the following source code into the Arduino IDE and save the file as "joystick.ino".
   Note: The following source code is provided only in this document.

```
#include "HardwareSerial.h"
HardwareSerial Serial;

int X = A5; //VCP A5 pin
int Y = A4; //VCP A4 pin
int SW = 2; //VCP 2 pin

void setup()
{
  pinMode(SW, INPUT_PULLUP);
  pinMode(X, INPUT);
  pinMode(Y, INPUT);
  Serial.begin(9600);
}

void loop()
{
  Serial.print(" Switch: ");
  Serial.print(digitalRead(SW));
  Serial.print(" X: ");
  Serial.print(analogRead(X));
  Serial.print(" Y: ");
  Serial.println(analogRead(Y));
  delay(100);
}
```

2. Verify and upload the "joystick.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include `joystick.ino.rom`.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\1D003A72AB3391D6D60FD7993380F8CD/joystick.ino.rom

[main:155] Complete FWDN
```

7.3 Gas Sensor

This example demonstrates how the VCP-G board uses the gas sensor (MQ 135) to detect various harmful gases in the air. It reads the analog value from a sensor connected to the analog pin on the VCP-G board, converts it to a voltage, and then prints the value to the serial monitor with one decimal place.

Note: Gas Sensor (MQ-135) is a product of Winsen®. All rights to its design, trademark, and related intellectual property are owned by Winsen.

7.3.1 Hardware Requirements

- VCP-G Board (x1)
- Gas sensor (MQ135) (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-socket Jumper Wires (x3)

7.3.2 Circuit

- Gas sensor
  - A0 pin of the gas sensor is connected to the analog pin A5 on the VCP-G board.
  - VCC pin of the gas sensor is connected to 5V on the VCP-G board.
  - GND pin of the gas sensor is connected to GND on the VCP-G board.

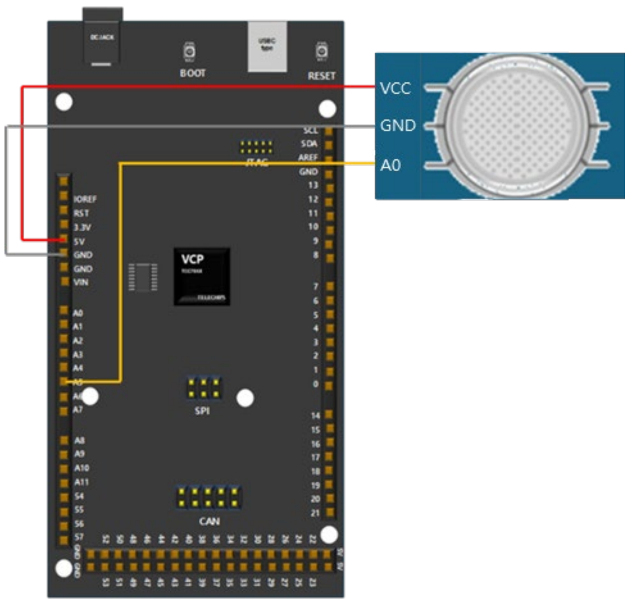

Figure 7.4 Gas Sensor Circuit Schematic

7.3.2.1 Pin Mapping

Table 7.4 describes the pin mapping of Gas Sensor.

Table 7.4 Pin Mapping of Gas Sensor

Pin Name	VCP-G Board	Arduino IDE
Gas Sensor A0 pin	A5	A5
Gas Sensor VCC pin	5V	-
Gas Sensor GND pin	GND	-

7.3.3 How to execute

1. Copy the following source code into the Arduino IDE and save the file as "GasSensor.ino".
   Note: The following source code is provided only in this document.

```
#include "HardwareSerial.h"
HardwareSerial Serial;

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

void loop()
{
  float vol;
  int sensorValue = analogRead(A5);
  vol=(float)sensorValue/1024*5.0;
  Serial.print(vol, 1);
 }
```

2. Verify and upload the "GasSensor.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
   If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include **`GasSensor.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\169CA3171ACB2F21DA0454D4A9F2FE50/GasSensor.ino.rom

[main:155] Complete FWDN
```

7.4 Metal Touch Sensor Module

This example demonstrates how the VCP-G board controls touch sensor and an LED on the breadboard. The metal touch sensor module (KY-036) is a versatile analog/digital sensor designed to detect touch on metal surfaces or human skin. This module uses a transistor to sense changes in electrical conductivity when touched, and it outputs both digital and analog signals for interaction with the VCP-G board.

When a touch is detected, the metal touch sensor module outputs relevant digital/analog values to the serial monitor. You can also control an LED based on the touch status.

Note: Metal Touch Sensor Module (KY-036) has a built-in potentiometer for adjusting sensitivity. You can turn this potentiometer to increase or decrease the sensitivity.

7.4.1 Hardware Requirements

- VCP-G Board (x1)
- Breadboard (x1)
- Metal Touch sensor module (KY-036) (x1)
- LED (x1)
- 220Ω Resistor (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x4)
- Pin-to-socket Jumper Wires (x4)

7.4.2 Circuit

- Metal Touch Sensor Module
  - A0 pin of the Metal Touch Sensor Module is connected to the analog pin A5 on the VCP-G board.
  - G pin of the Metal Touch Sensor Module is connected to GND on the VCP-G board.
  - (+) pin of the Metal Touch Sensor Module is connected to 5V on the VCP-G board.
  - D0 pin of the Metal Touch Sensor Module is connected to pin 30 on the VCP-G board.

- LED
  - (+) pin of the LED is connected to pin 13 on the VCP-G board.
  - (-) pin of the LED is connected to GND on the VCP-G board with a 220Ω resistor.

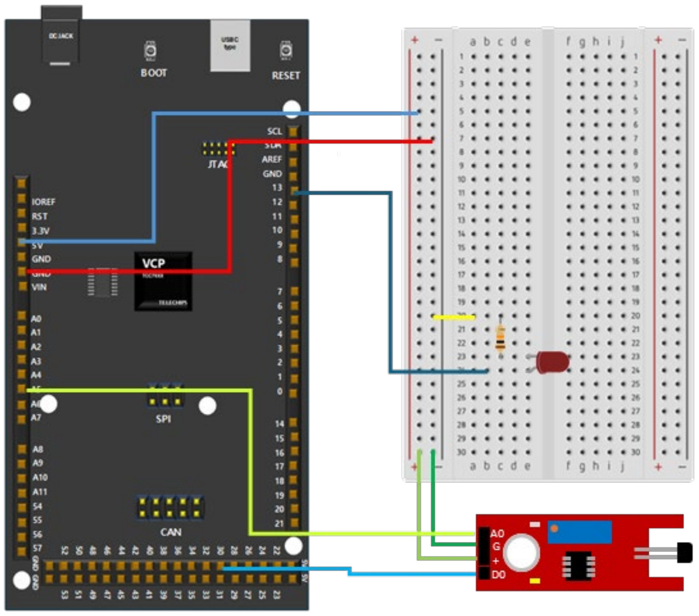

Figure 7.5 Metal Touch Sensor Circuit Schematic

7.4.2.1 Pin Mapping

Table 7.5 describes the pin mapping of Metal Touch Sensor.

Table 7.5 Pin Mapping of Metal Touch Sensor

Pin Name	VCP-G Board	Arduino IDE
Metal Touch sensor module A0 pin	A5	A5
Metal Touch sensor module G pin	GND	-
Metal Touch sensor module (+) pin	5V	-
Metal Touch sensor module D0 pin	30	30
LED (+) pin	13	13
LED (-) pin	GND	-

7.4.3 How to execute

1. Copy the following source code into the Arduino IDE and save the file as "vcp_touch.ino".
   Note: The following source code is provided only in this document.

```
#include <Arduino.h>
#include "HardwareSerial.h"
HardwareSerial Serial;

const int touchPin = 30;   // touch sensor D0 pin - VCP 30 pin
const int ledPin = 13;     // LED - VCP 13 pin
const int analogPin = A5; // touch sensor A0 pin - VCP A5 pin

int touchState = 0;    // Variable to store the touch sensor state
int touchIntensity = 0; // Variable to store the touch intensity

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

void loop() {
  touchState = digitalRead(touchPin);
  touchIntensity = analogRead(analogPin); // Read the analog value from the sensor

  if (touchState == HIGH) {
    Serial.print("Touch detected, ");  // Print message when touch is detected
    digitalWrite(ledPin, HIGH);       // LED turns on when touch is detected
  }
  else {
    Serial.print("No touch detected, ");  // Print message when touch is not detected
    digitalWrite(ledPin, LOW);           // LED turns off when touch is not detected
    touchIntensity = 0;                 // Set touch intensity to 0 when no touch is detected
  }

  Serial.print("Touch intensity: ");
  Serial.println(touchIntensity);  // Print the touch intensity value

  delay(500);
}
```

2. Verify and upload the "vcp_touch.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
   If the issue persists, try running the Arduino IDE with administrator privileges.

4. After successfully uploading the file, check the Arduino IDE output console for the following message.
   Note: The message should include **`vcp_touch.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\567805554B5B9F915DCC80B38483AE07/vcp_touch.ino.rom

[main:155] Complete FWDN
```

Note: Set an appropriate baud rate for serial communication.

7.5 Step Motor with Motor Driver

This example demonstrates how the VCP-G board controls 4-wire stepper motor (28BYJ-48 (5V DC)) and Motor Driver (ULN2003 (5V–12V)). The code in Chapter 7.5.3 defines the pins connected to the motor driver and sets the number of steps per revolution. The motor rotates forward for one full revolution, pauses, then rotates backward for one full revolution, and pauses again. The motor speed is controlled by the delay between steps, and the direction is controlled by the order in which the coils are activated.

Note: The 28BYJ-48 motor requires 4096 signals for one full rotation in half step mode and 2048 signals for one full rotation in full step mode. For precise motor control, the number of signals required depending on the mode should be considered.

7.5.1 Hardware Requirements

- VCP-G Board (x1)
- Step Motor (28BYJ-48) (x1)
- Motor Driver (ULN2003) (x1)
- 12V 1A Power Adapter (x1)
- USB Type-C to A Cable (x1)
- Pin-to-pin Jumper Wires (x6)

7.5.2 Circuit

- Motor Driver
  - IN1 pin is connected to pin 8 on the VCP-G board.
  - IN2 pin is connected to pin 9 on the VCP-G board.
  - IN3 pin is connected to pin 10 on the VCP-G board.
  - IN4 pin is connected to pin 11 on the VCP-G board.
  - (+) pin is connected to 5V on the VCP-G board.
  - (-) pin is connected to GND on the VCP-G board.

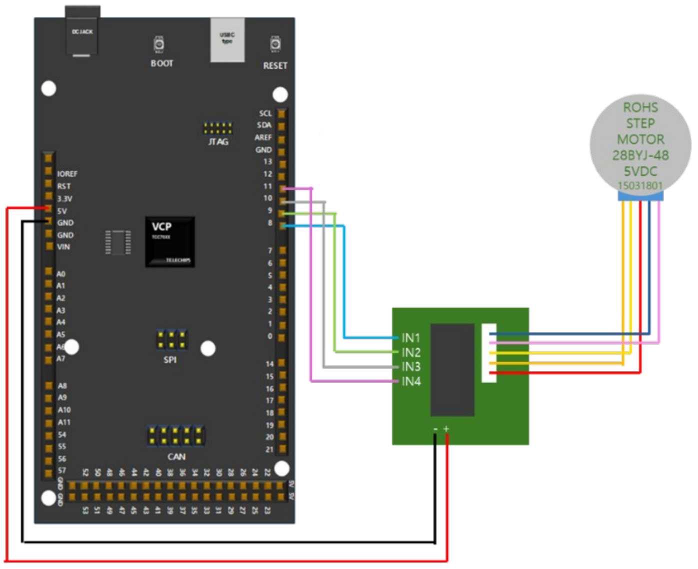

Figure 7.6 Step Motor with Motor Driver Circuit Schematic

7.5.2.1 Pin Mapping

Table 7.6 describes the pin mapping of Motor Driver.

Table 7.6 Pin Mapping of Motor Driver

Pin Name	VCP-G Board	Arduino IDE
Motor Driver IN1 pin	8	8
Motor Driver IN2 pin	9	9
Motor Driver IN3 pin	10	10
Motor Driver IN4 pin	11	11
Motor Driver (+) pin	5V	-
Motor Driver (-) pin	GND	-

7.5.3 How to execute

1. Copy the following source code into the Arduino IDE and save the file as "motordriver.ino".
   Note: The following source code is provided only in this document.

```
/*
  This code controls a 4-wire stepper motor using an Arduino.
  It defines the pins connected to the motor driver and sets the number of steps per revolution.
  The motor rotates forward for one full revolution, pauses, then rotates backward for one full revolution,
and pauses again.
  The motor's speed is controlled by the delay between steps, and the direction is controlled by the order
in which the coils are activated.
*/

#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 11

const int stepsPerRevolution = 2037; // Number of steps per revolution

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop() {
  // Rotate forward
  for (int i = 0; i < stepsPerRevolution; i++) {
    stepMotor(i % 4); // Call stepMotor with the current step (0 to 3)
    delay(2); // Adjust speed
  }
  delay(1000);

  // Rotate backward
  for (int i = 0; i < stepsPerRevolution; i++) {
    stepMotor(3 - (i % 4)); // Call stepMotor with the current step in reverse order (3 to 0)
    delay(2);
  }
  delay(1000);
}

void stepMotor(int step) {
  switch (step) {
    case 0:
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      break;
    case 1:
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      break;
    case 2:
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);
      break;
    case 3:
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);
      break;
  }
}
```

2. Verify and upload the "motordriver.ino" file to the VCP-G board.
3. If the upload process gets stuck in an infinite uploading state, it is because the FWDN mode is not activated. To resolve this:
   1) Disconnect the power cable from the VCP-G board.
   2) Press and hold the FWDN button.
   3) Reconnect the power cable while continuing to hold the FWDN button.
   4) Release the FWDN button.
      If the issue persists, try running the Arduino IDE with administrator privileges.
4. After successfully uploading the file, check the Arduino IDE output console for the following message:
   Note: The message should include **`motordriver.ino.rom`**.

```
[FWDN_VCP::WriteFile:577] Complete to send file -
C:\Users\topst\AppData\Local\arduino\sketches\1D003A72AB3391D6D60FD7993380F8CD/motordriver.ino.rom

[main:155] Complete FWDN
```

8 REFERENCES

[1] Contact TOPST for more details: topst@topst.ai

Note: Reference documents can be provided whenever available, depending on the terms of a contract. If the reference documents are unavailable, the contents directly related to your development can be guided.

9 REVISION HISTORY

Rev. 1.01: 2025-07-18

- Updated
  - Chapter 1: Description
  - Chapter 2: Changed chapter title from “VCP Digital” to “VCP-G Digital Pins”
  - Chapter 2.1.1: Description
  - Chapter 2.1.2:
    - Description
    - Figure 2.1
  - Chapter 2.2.1: Description
  - Chapter 2.2.2:
    - Description
    - Figure 2.2
  - Chapter 3: Changed chapter title from “VCP Analog” to “VCP-G Analog”
  - Chapter 3.1.3: Description
  - Chapter 4: Changed chapter title from “VCP SPI” to “VCP-G SPI”
  - Chapter 4.1.3:
    - Added **Note**
    - Table 4.2
  - Chapter 5: Changed chapter title from “VCP I2C” to “VCP-G I2C”
  - Chapter 6: Changed chapter title from "VCP UART” to “VCP-G UART”
  - Chapter 7.3.3: Description
  - Chapter 7.4.1: Description

- Deleted
  - Chapter 7.2.2.2: Circuit Schematic
  - Chapter 7.4.2.2: Circuit Schematic

Rev. 1.00: 2025-03-07

- Official version release