My Docs

None

Qijun Han

None

Table of contents

1. Welcome to MkDocs

For full documentation visit mkdocs.org.

1.1 Commands

- `mkdocs new [dir-name]` - Create a new project.
- `mkdocs serve` - Start the live-reloading docs server.
- `mkdocs build` - Build the documentation site.
- `mkdocs -h` - Print help message and exit.

1.2 Project layout

```
mkdocs.yml    # The configuration file.
docs/
    index.md  # The documentation homepage.
    ...       # Other markdown pages, images and other files.
```

2. Lab Reports

2.1 Lab1 Raspberry Pi SSH VNC

2.1.1

SD Raspberry Pi Imager SSH VNC SD

2.1.2

- SD Formmater SD Raspberry Pi Imager Win32DiskImager SD
- SSH Secure Shell
- VNC Virtual Network Computing VNC Viewer VNC

2.1.3

- SD Formmater SD
- Raspberry Pi Imager
- SD config.txt
- SD
-
- SSH VNC IP
- Putty MobaXterm SSH
- VNC Viewer xrdp Windows

2.2 Lab 2

Lab2 LED

Raspberry Pi IO wiringPi RPi.GPIO Mu Geany IDE Python C/C++ LED

1. **Raspberry Pi IO**
2. Raspberry Pi 40 GPIO wiringPi BCM2837 SOC
3. BCM
4. **wiringPi**
5. wiringPi C/C++ GPIO Raspberry Pi GPIO GPIO
6. **RPi.GPIO**
7. RPi.GPIO Python Python Raspberry Pi GPIO Raspbian API
8. **Mu Geany IDE**
9. Mu Python IDE
10. Geany IDE C/C++
11. **LED**
12. LED LED

1.
2. LED S Raspberry Pi GPIO GND Raspberry Pi GPIO BCM GPIO19, GPIO20, GND
3.
4. Mu Python LED GPIO / LED
5. C/C++ Geany GPIO wiringPi
6.
7. Python C/C++ LED
8.
9.
10.

2.3 Lab 3

Lab3

Raspberry Pi GPIO LED

1.
2. S 0V 3.3V 5V
3. **GPIO**
4. Raspberry Pi GPIO GPIO
5. **LED**
6. LED
7.
8. RC

1.
2. SIG(S) Raspberry Pi GPIO17 VCC 5V GND LED
3. LED T GPIO19, GPIO20, GND LED GPIO27 GND
4.
5. Python RPi.GPIO GPIO
6. GPIO17
7.
8. LED
9.
10. Python LED
11. LED
12.
13.
14.
15.
16. GPIO

2.4 Lab 4

Lab5 PCF8591

PCF8591 Analog-to-Digital Converter, ADC Raspberry Pi PCF8591 8 ADC I2C PCF8591 LED

1. **PCF8591**
2. PCF8591 CMOS 8 A/D 8 D/A
3. I2C 0x48 A0, A1, A2 8 I2C
4. **I2C**
5. I2C SDA SCL Raspberry Pi PCF8591
6.
7. AIN0 AOUT LED LED
8. NTC

1.
2. Raspberry Pi T PCF8591 SDA SCL VCC GND
3. LED PCF8591 AOUT GND LED GND GPIO GPIO17
4. **I2C**
5. Raspberry Pi Preferences -> Raspberry Pi Configuration
6. Interfaces I2C OK
7.
8. Python `smbus` I2C
9. SMBus PCF8591 AIN0 AOUT LED
10.

```
import smbus
import time

# Define the I2C address of the PCF8591 and control bits
address = 0x48  # Default address for PCF8591
control_bit = 0x40  # Command to start conversion on channel 0 (AIN0)

# Initialize the SMBus library
bus = smbus.SMBus(1)  # Use I2C bus 1

try:
    while True:
        # Write the control byte to initiate an A/D conversion on channel 0
        bus.write_byte(address, control_bit)

        # Read back the converted value from the PCF8591
        analog_value = bus.read_byte(address)

        # Print out the raw analog value
        print("Analog Value:", analog_value)

        # Map the analog value to a range suitable for controlling LED brightness
        led_brightness = int((analog_value / 255.0) * 100)

        # Here you would add code to set the LED brightness using PWM or similar method.
        # For demonstration purposes, we'll just print the calculated brightness.
        print("LED Brightness (%):", led_brightness)

        time.sleep(0.1)  # Small delay between readings
```

```
except KeyboardInterrupt:
    pass  # Allow the program to exit cleanly with Ctrl+C
```

1.
2. Python LED
3.
4.
5.

2.5 Lab 5

NTC Raspberry Pi PCF8591
Raspberry Pi /

NTC NTC Steinhart-Hart
- NTC 10kΩ PCF8591
AIN0 PCF8591 A/D 5V ADC 8 0~255 0~5V
Steinhart-Hart T Steinhart-Hart Steinhart-Hart 1 T = A + B ln (R) + C (ln (R)) 3 T
1 =A+Bln(R)+C(ln(R)) 3 T T R R A A, B B, C C R 0 R 0 10kΩ B
3950K Raspberry Pi T PCF8591 SDA SCL VCC GND AO PCF8591
AIN0 DO VCC 5V GND I2C Raspberry Pi Preferences -> Raspberry Pi Configuration
Interfaces I2C OK Python smbus I2C SMBus PCF8591
AIN0

```
import smbus
import math
import time

# Define the I2C address of the PCF8591 and control bits
address = 0x48  # Default address for PCF8591
control_bit = 0x40  # Command to start conversion on channel 0 (AIN0)

# Constants for the thermistor calculation
R0 = 10000  # Resistance at 25°C in ohms
B = 3950  # Thermistor constant in Kelvin
T0 = 298.15  # Standard temperature in Kelvin (25°C)
Vcc = 5.0  # Supply voltage in volts

# Initialize the SMBus library
bus = smbus.SMBus(1)  # Use I2C bus 1

def read_temperature():
    try:
        # Write the control byte to initiate an A/D conversion on channel 0
        bus.write_byte(address, control_bit)

        # Read back the converted value from the PCF8591
        analog_value = bus.read_byte(address)

        # Calculate the analog voltage
        Vr = (analog_value / 255.0) * Vcc

        # Calculate the resistance of the thermistor
        Rt = R0 * Vr / (Vcc - Vr)

        # Apply the Steinhart-Hart equation to calculate temperature
        temp_kelvin = 1 / (math.log(Rt / R0) / B + 1 / T0)
        temp_celsius = temp_kelvin - 273.15

        return round(temp_celsius, 2)

    except Exception as e:
        print("Error reading temperature:", str(e))
        return None

try:
    while True:
        temperature = read_temperature()
        if temperature is not None:
            print(f"Temperature: {temperature}°C")
        else:
            print("Failed to read temperature.")

        time.sleep(1)  # Small delay between readings

except KeyboardInterrupt:
    pass  # Allow the program to exit cleanly with Ctrl+C
```

Python

2.6 Lab 6

2.6.1

2.6.2

2.6.3

Lab6

HC-SR04 Raspberry Pi HC-SR04 Python

1.
2. Trig 10 8 40kHz
3. Echo Echo
4. 343 / 20 Echo
5.
6. **VCC** 5V
7. **Trig**
8. **Echo**
9. **GND**
10.
11. GPIO 3.3V Echo 5V Echo

1.
2. Raspberry Pi T VCC Trig Echo GND
3. Trig Raspberry Pi GPIO17 BCM Echo GPIO18 BCM VCC 5V GND
4.
5. Python RPi.GPIO GPIO
6. `get_distance()` Trig 10 Echo Echo
7.

```
import RPi.GPIO as GPIO
import time

# Define GPIO pins for the ultrasonic sensor
TRIG = 17  # BCM numbering
ECHO = 18  # BCM numbering

# Setup GPIO mode and pin directions
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

def get_distance():
    # Ensure TRIG is low initially
    GPIO.output(TRIG, False)
    time.sleep(0.2)

    # Send a 10us pulse to TRIG
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    # Wait for ECHO to go high
    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()

    # Wait for ECHO to go low again
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    # Calculate the duration of the pulse
    pulse_duration = pulse_end - pulse_start

    # Convert pulse duration to distance in centimeters
    distance = pulse_duration * 17150  # Speed of sound in cm/s divided by 2 (round trip)
    distance = round(distance, 2)

    return distance

try:
    print("Measuring distance...")
    while True:
        dist = get_distance()
        print(f"Distance: {dist} cm")
        time.sleep(1)

except KeyboardInterrupt:
    print("Measurement stopped by user")

finally:
    GPIO.cleanup()  # Clean up GPIO settings before exiting
```

1.
2. Python
3.
4.
5.

2.7 Lab 7

2.8 Lab 8

2.8.1

2.8.2

2.8.3

2.9 Lab 9

2.9.1

2.9.2

2.9.3

2.10 Lab 10

2.10.1

2.10.2

- `add_event_detect()` GPIO
- `wait_for_edge()` CPU

2.10.3

LED GPIO LED