HappyFish

Sensors & LED Manual

Step-by-Step Instructions for pH, Temperature, and TDS Sensors
with LED Sunrise and Sunset Simulation

CUNY Spring Forward 2024

Parthkumar Joshi

Table of Contents

Introduction

Welcome to the "Happyfish Sensor and Lighting Control Manual." This guide provides step-by-step instructions on how to use specific python files required to operate pH, temperature, and TDS sensors, as well as how to manage LEDs to simulate sunrise and sunset cycles. Designed as a quick reference, this manual ensures you have the necessary information to effectively run and maintain these components. Follow the detailed procedures outlined here to efficiently operate the sensors.

Requirements

Before running any of the programs, please ensure that you have:

1. **Enabled I2C and One-Wire on your Raspberry Pi**:
   - Go to the Raspberry Pi configuration by typing sudo raspi-config in the terminal.
   - Navigate to "Interfacing Options" and select "I2C" and "One-Wire" to enable them.
   - Reboot your Raspberry Pi after making these changes.
2. **Verified the I2C addresses**:
   - Use the command i2cdetect -y 1 to detect the I2C addresses of your devices.
   - Ensure that the I2C addresses in the code match the detected addresses.
3. **Configured the ADS1115 switch**:
   - Make sure the switch on the ADS1115 is set to the correct address (0x49 or 0x48).
4. **Calibrate Sensors**:
   - Calibrate the pH sensor using the program before testing.
5. **Used the correct GPIO pins**:
   - Verify that you are using the correct GPIO pins for your pH sensor, TDS sensor, and temperature sensor.

Temperature Sensor

1. `temp.py`

   temp.py is designed to read and print temperature measurements from a DS18B20 sensor connected to a Raspberry Pi. The script initializes the necessary kernel modules, identifies the device folder, and sets the file path for the temperature sensor data. It then defines two functions: read_temp_raw() to read the temperature sensor data and read_temp() to process the data and extract the temperature value. The script enters an infinite loop, continuously reading and printing the temperature in both Celsius and Fahrenheit every 5 seconds.

Hardware Requirements:

- Raspberry Pi
- Temperature sensor (DS18B20)
- Jumper wires (for connecting to the Raspberry Pi)

Hardware Setup:

- Connect the temperature sensor to the Raspberry Pi as follows:
  - VCC pin of the sensor to pin 1 (3.3V) on the Raspberry Pi
  - GND pin of the sensor to pin 6 (GND) on the Raspberry Pi
  - DATA pin of the sensor to pin 7 (GPIO 4) on the Raspberry Pi
- Ensure the jumper wires are securely connected to both the sensor and the Raspberry Pi

Software Requirements:

- The w1-gpio and w1-therm kernel modules should be loaded (this is done automatically in the code)

How to Run:

1. Edit the /boot/config.txt file using sudo nano /boot/config.txt and add the following lines at the end of the file:
   - dtoverlay=w1-gpio
   - dtoverlay=w1-therm

   Save and exit the editor. Reboot the Raspberry Pi.
2. Open a terminal and navigate to the directory where you saved the file
3. Run the file using Python: python temp.py
4. The script will continuously print the temperature in Celsius and Fahrenheit every 5 seconds.

pH Sensor

1. DFRobot_PH.py

   DFRobot_PH.py is a class definition for interfacing with a pH sensor. The class, DFRobot_PH, provides methods for initializing the pH sensor, reading pH values, calibrating the sensor using buffer solutions, and resetting the calibration data to default values. The begin method initializes the pH sensor by reading the acid and neutral voltage values from a file named phdata.txt. If the file does not exist, the method prints an error message and exits. The read_PH method converts a voltage value to a pH value with temperature compensation.The calibration method calibrates the pH sensor using buffer solutions of pH 7.0 and 4.0. It writes the corresponding voltage values to the phdata.txt file. The reset method resets the calibration data to default values, writing the default acid and neutral voltage values to the phdata.txt file. This class serves as a foundation for other pH-related files, providing a way to interact with the pH sensor and perform calibration and pH value conversions.

2. demo_PH_reset.py

demo_PH_reset.py is a program designed to reset a pH sensor. It imports the DFRobot_PH class from the DFRobot_PH module and creates an instance of the class. The script then calls the reset method of the class to reset the pH sensor. After a short delay of 0.5 seconds, the script exits with a status code of 1, indicating successful execution. The primary function of this script is to reset the pH sensor in preparation for pH measurements.

Hardware Requirements:

- Raspberry Pi
- pH sensor

Software Requirements:

- DFRobot_PH module (DFRobot_PH.py)

Running the File:

1. Make sure to have the DFRobot_PH.py file in the same directory.
2. Connect the pH sensor to the Raspberry Pi with the ADS1115.
3. Run the file using Python: python3 demo_PH_reset.py.
4. The script will reset the pH sensor and exit with a status code of 1.

3. demo_PH_EC.py

demo_PH_EC.py is a program to read pH values from a pH sensor and EC values from a TDS sensor, using real-time temperature readings from a temperature sensor (DS18B20). The script initializes the ADS1115 analog-to-digital converter (ADC) module, the DFRobot_EC module for EC readings, and the DFRobot_PH module for pH readings. It then enters an infinite loop, where it reads temperature values from the temperature sensor, voltage values from the pH and TDS sensors using the ADS1115 and converts these voltage values to pH and EC values using the readEC and read_PH methods of the DFRobot_EC and DFRobot_PH classes, respectively. The script also performs temperature compensation for the pH and EC readings. The temperature, EC, and pH values are then printed to the console, and the loop waits for 1 second before taking the next reading.

Hardware Requirements:

- Raspberry Pi (any version)
- ADS1115
- pH Sensor
- Temperature sensor (DS18B20)
- Jumper wires and breadboard for connections

Hardware Connections:

- ADS1115 module:
    - VCC to Raspberry Pi's 3.3V pin
    - GND to Raspberry Pi's GND pin
    - SCL to Raspberry Pi's I2C clock pin (SCL)
    - SDA to Raspberry Pi's I2C data pin (SDA)
- TDS Sensor:
    - Connected to ADS1115 module's analog input (channel 0)

- DFRobot_PH module:

    - Connected to ADS1115 module's analog input (channel 1)
- Temperature sensor (DS18B20):
    - VCC to Raspberry Pi's 3.3V pin
    - GND to Raspberry Pi's GND pin
    - Data to Raspberry Pi's GPIO pin (GPIO 4)

Software Requirements:

- Imported files: CQRobot_ADS1115.py, DFRobot_EC.py, and DFRobot_PH.py

Setup and Running the Script:

1. Connect the ADS1115 module to the Raspberry Pi's I2C pins.
2. Connect the TDS Sensor to the ADS1115 module's analog input (channel 0)
3. Connect the pH sensor to the ADS1115 module's analog input (channel 1).
4. Connect the temperature sensor to the Raspberry Pi.
5. Place the Python script in a directory with the imported modules.
6. Run the Python script using python script_name.py.
7. The script will read and print the temperature, EC, and pH values.

4. demo_PH_calibration.py

demo_PH_calibration.py is a program to calibrate a pH sensor using two buffer solutions as base lines (pH 4 and 7). The script initializes the ADS1115 analog-to-digital converter (ADC) module and the DFRobot_PH module, which interfaces with the pH sensor. It then enters a calibration loop, where it reads voltage values from the pH sensor using the ADS1115 and performs calibration using the calibration method of the DFRobot_PH class. The script limits the calibration loop to a specified number of iterations. After completing the calibration process, it prints a message indicating the number of calibration cycles completed.

Hardware Requirements:

- Raspberry Pi
- Buffer Solutions (pH 4/7)
- pH sensor

Software Requirements:

- ADS1115 module (CQRobot_ADS1115.py)
- DFRobot_PH module (DFRobot_PH.py)

Running the File:

1. Make sure to have the CQRobot_ADS1115.py file and DFRobot_PH.py file in the same directory.
2. Connect the ADS1115, pH sensor to the Raspberry Pi.
3. Run the file using Python: python3 demo_PH_calibration.py.
4. The script will perform pH calibration using the ADS1115 module and pH sensor.
5. The script will print the voltage values and calibration iterations to the console for 5 loops.

5. demo_PH_read.py

demo_PH_read.py is a program to read pH values from a pH sensor. The script initializes the ADS1115 analog-to-digital converter (ADC) module and the DFRobot_PH module, which interfaces with the pH sensor. It then enters an infinite loop, where it reads voltage values from the pH sensor using the ADS1115 and converts the voltage to a pH value using the read_PH method of the DFRobot_PH class. The script also assumes a fixed temperature value of 25 degrees Celsius for temperature compensation. The pH value is then printed to the console along with the temperature, and the loop waits for 1 second before taking the next reading.

Hardware Requirements:

- Raspberry Pi
- ADS1115
- pH Sensor

Software Requirements:

- The CQRobot_ADS1115.py and DFRobot_PH.py files must be in the same directory as this Python file.

Running the File:

1. Connect the pH sensor to the ADS1115 module:
   - Connect the pH sensor's analog output to the ADS1115's A0 channel (channel 0).
   - Connect the pH sensor's ground to the ADS1115's GND pin.
2. Connect the ADS1115 module to the Raspberry Pi:
   - Connect the ADS1115's I2C SCL pin to the Raspberry Pi's I2C clock pin (SCL).
   - Connect the ADS1115's I2C SDA pin to the Raspberry Pi's I2C data pin (SDA).

- Connect the ADS1115's VCC pin to the Raspberry Pi's 3.3V power pin.
- Connect the ADS1115's GND pin to the Raspberry Pi's GND pin.

3. Place the CQRobot_ADS1115.py and DFRobot_PH.py files in the same directory as this Python file.
4. Run the file using Python: python3 demo_PH_read.py
5. The script will continuously read the pH values and print them to the console.

6. demo_PH_read.py

demo_PH_read.py is a program to read pH values from a pH sensor. The script initializes the ADS1115 analog-to-digital converter (ADC) module and the DFRobot_PH module, which interfaces with the pH sensor. It then enters an infinite loop, where it reads voltage values from the pH sensor using the ADS1115 and converts the voltage to a pH value using the read_PH method of the DFRobot_PH class. The script also assumes a fixed temperature value of 25 degrees Celsius for temperature compensation. The pH value is then printed to the console along with the temperature, and the loop waits for 1 second before taking the next reading.

Hardware Requirements:

- Raspberry Pi
- ADS1115
- pH Sensor

Software Requirements:

- The CQRobot_ADS1115.py and DFRobot_PH.py files must be in the same directory as this Python file.

Running the File:

1. Connect the pH sensor to the ADS1115 module:
   - Connect the pH sensor's analog output to the ADS1115's A0 channel (channel 0).
   - Connect the pH sensor's ground to the ADS1115's GND pin.
2. Connect the ADS1115 module to the Raspberry Pi:
   - Connect the ADS1115's I2C SCL pin to the Raspberry Pi's I2C clock pin (SCL).
   - Connect the ADS1115's I2C SDA pin to the Raspberry Pi's I2C data pin (SDA).

- Connect the ADS1115's VCC pin to the Raspberry Pi's 3.3V power pin.
- Connect the ADS1115's GND pin to the Raspberry Pi's GND pin.

3. Place the CQRobot_ADS1115.py and DFRobot_PH.py files in the same directory as this Python file.
4. Run the file using Python: python3 demo_PH_read.py
5. The script will continuously read the pH values and print them to the console.

Note: For all programs relating to pH sensor, make sure to have ecdata.txt and phdata.txt in the same directory.

TDS Sensor

1. CQRobot_ADS1115.py

   CQRobot_ADS1115.py is a driver for the ADS1115 analog-to-digital converter (ADC) integrated circuit. The script provides a class, ADS1115, that allows users to interact with the ADS1115 device connected to an I2C bus. The class offers methods to set the gain, address, channel, and configuration of the ADS1115, as well as read voltage and comparator values from the device. The script also defines constants for the ADS1115's register map and configuration options. The primary functions of the script are to configure and read data from the ADS1115 ADC, which can be used in a variety of applications, such as measuring voltage levels, monitoring sensor outputs, or controlling external devices.

2. ADS1115_ReadVoltage.py

   ADS1115_ReadVoltage.py is a program designed to read and process voltage measurements from a TDS (Total Dissolved Solids) sensor connected to an ADS1115 analog-to-digital converter (ADC). The script initializes the ADS1115 module, sets its I2C address and gain, and continuously takes voltage readings from the sensor. It uses a median filter to smooth out the readings and calculates the TDS value based on the voltage readings and a compensation coefficient that accounts for temperature variations. The script then prints the TDS value in parts per million (ppm) to the console at regular intervals.

Hardware Requirements:

- Raspberry Pi
- ADS1115 module
- TDS sensor (connected to ADS1115 channel A1)

Software Requirements:

- The CQRobot_ADS1115.py file containing the ADS1115 class must be in the same directory as this Python file.

Running the File:

1. Connect the TDS sensor to the ADS1115 module's A1 channel.
2. Connect the ADS1115 module to the Raspberry Pi:
    - Connect the ADS1115's I2C SCL pin to the Raspberry Pi's I2C clock pin (SCL).
    - Connect the ADS1115's I2C SDA pin to the Raspberry Pi's I2C data pin (SDA).
    - Connect the ADS1115's VCC pin to the Raspberry Pi's 3.3V power pin.
    - Connect the ADS1115's GND pin to the Raspberry Pi's GND pin.
3. Place the CQRobot_ADS1115.py file in the same directory as this Python file.
4. First run CQRobot_ADS1115: python3 CQRobot_ADS1115.py
5. Then run the file using Python: python3 ADS1115_ReadVoltage.py
6. The script will continuously read the TDS values and print them to the console.

Note: After thorough testing, more accurate TDS readings are produced when sensor is connected to the Adafruit cobbler. Calibration process still needs to be found/made for TDS sensor.

LEDs

1. Light_Controller.py

   This code is a script that controls the brightness of LEDs using a PCA9685 PWM driver. It defines a Schedule class that determines the current stage of the day based on the current time and the configured sunrise and sunset times.
   The Schedule class has three stages: pre_sun_rise, sun_rise, lights_on, sun_set, and post_sun_set. The stage is determined based on the current time and the configured sunrise and sunset times.The script calculates the brightness per centage based on the current stage. For example, during sun_rise and sun_set, the brightness increases or decreases gradually. The script uses the PCA9685 library to control the PWM driver, which sets the brightness of the LEDs. The script prints the current time, stage, and brightness level every minute. The script runs indefinitely until it's manually stopped. In summary, this code is designed to simulate a day-night cycle for LEDs, adjusting their brightness based on the time of day.

Hardware Requirements:

- Raspberry Pi
- PCA9685 16-channel PWM driver
  - Connect VCC to Raspberry Pi's 3.3V pin
  - Connect GND to Raspberry Pi's GND pin
  - Connect SCL to Raspberry Pi's GPIO 5 (I2C clock)
  - Connect SDA to Raspberry Pi's GPIO 6 (I2C data)
- LEDs connected to the PCA9685

    - Connect LED or light Positive leg to PCA9685 output channel (e.g., Channel 0)
    - Connect LED or light Negative leg to GND

Software Requirements:

- adafruit_pca9685 library installed

Steps to Run:

1. Connect the PCA9685 to the Raspberry Pi's I2C pins (SCL and SDA).
2. Connect the LEDs or lights to the PCA9685's output channels.
3. Install the adafruit_pca9685 library using pip: sudo pip3 install adafruit-circuitpython-pca9685
4. Set the environment variables SUNRISE, SUNSET, and DURATION to the desired values.
5. Run the script using Python: python3 light_controller.py
6. The script will continuously log the current time, stage, and brightness level, and adjust the PWM duty cycle to control the LEDs or lights.

Note: Make sure to adjust the SUNRISE, SUNSET, and DURATION environment variables to match your specific requirements.

2. sun_cycle_simulation.py

The sun_cycle_simulation.py script simulates sunrise and sunset using LEDs bycontrolling their brightness. It utilizes the adafruit_pca9685 library to interact with a PCA9685 PWM driver, which controls the brightness of the LEDs. The script defines several functions, including set_pwm to set the PWM duty cycle for a single channel, brightness_percentage to calculate and return the brightness percentage, and adjust_leds_brightness to gradually adjust the brightness for all 12 LEDs over a specified duration. In the interactive control section, the script provides a while loop that continuously prompts the user to enter 'sunrise', 'sunset', or 'exit'. If the user enters 'sunrise', the script simulates a sunrise by gradually increasing the brightness of the LEDs over the specified duration. If the user enters 'sunset', the script simulates a sunset by gradually decreasing the brightness of the LEDs over the specified duration. If the user enters 'exit', the script exits the program. The program can be stopped manually using Ctrl+C, and the PCA9685 PWM driver is cleaned up before exiting.

Hardware Requirements:

- Raspberry Pi
- PCA9685 16-channel PWM driver
  - Connect VCC to Raspberry Pi's 3.3V pin
  - Connect GND to Raspberry Pi's GND pin
  - Connect SCL to Raspberry Pi's GPIO 5 (I2C clock)
  - Connect SDA to Raspberry Pi's GPIO 6 (I2C data)
- 12 LEDs connected to PCA9685 output channels (0-11)
  - Connect LED Positive leg to PCA9685 output channel (e.g., Channel 0)

- Connect LED Negative leg to GND

Software Requirements:

- adafruit_pca9685 library installed

Steps to Run:

1. Connect the PCA9685 to the Raspberry Pi's I2C pins (SCL and SDA).
2. Connect the 12 LEDs to the PCA9685's output channels (0-11).
3. Install the adafruit_pca9685 library using pip: pip sudo pip3 install adafruit-circuitpython-pca9685
4. Run the script using Python: python3 sun_cycle_simulation.py
5. The script will interactively prompt the user to simulate a sunrise, sunset, or exit the program.

Manual Commands:

- To simulate a sunrise, enter sunrise and press Enter.
- To simulate a sunset, enter sunset and press Enter.
- To exit the program, enter exit and press Enter.

Temperature + pH + TDS Sensors

1. happyfishV4.py

   happyfishV4.py is a program designed to read pH, TDS (Total Dissolved Solids), and temperature values using individual sensors. The script starts by importing necessary libraries, including ADS1115 and DFRobot_PH, which are used to interact with the pH sensor. The script sets up the DS18B20 temperature sensor by loading the necessary kernel modules and setting the base directory for the temperature sensor. The script defines two functions to read the temperature sensor data: read_temp_raw reads the raw data from the temperature sensor and read_temp reads and processes the temperature data. The script then defines the ADS1115 configuration and creates ADS1115 objects for the pH and TDS sensors, setting their I2C addresses and gain values. The script creates a DFRobot_PH object, ph_sensor, and initializes it. In the main loop, the script continuously monitors and prints the real-time temperature, pH, and TDS values. For each iteration, it reads the real-time temperature, reads the pH value using the DFRobot_PH object, reads the TDS value using the ADS1115 object, and prints the results. The script continues running until it is stopped manually using Ctrl+C, at which point it prints a message indicating that the script execution was stopped by keyboard interruption.

Hardware Requirements:

- Raspberry Pi
- DS18B20 temperature sensor
  - Connect VCC to Raspberry Pi's 3.3V pin

    - Connect GND to Raspberry Pi's GND pin
    - Connect DATA to Raspberry Pi's GPIO 4
- ADS1115 analog-to-digital converter (I2C: 0x49)
    - Connect VCC to Raspberry Pi's 3.3V pin
    - Connect GND to Raspberry Pi's GND pin
    - Connect SCL to Raspberry Pi's GPIO 5 (I2C clock)
    - Connect SDA to Raspberry Pi's GPIO 6 (I2C data)
- pH sensor
    - Connect to ADS1115 channel 2 (for pH measurement)
- TDS sensor
    - Connect to ADS1115 channel 0 (for TDS measurement)

Software Requirements:

- CQRobot_ADS1115 and DFRobot_PH files installed in same directory

Steps to Run:

1. Connect the DS18B20 temperature sensor to the Raspberry Pi as described above.
2. Connect the ADS1115 to the Raspberry Pi as described above.
3. Connect the pH sensor to ADS1115 channel 2.
4. Connect the TDS sensor to ADS1115 channel 0.
5. CQRobot_ADS1115.py and DFRobot_PH.py files in same directory.
6. Run the script using Python: python3 happyfishV4.py
7. The script will continuously monitor and print the real-time temperature, pH, and TDS values.

Note: Make sure to adjust the I2C address and gain settings for the ADS1115 and sensors according to your specific setup. Can check I2C addresses detected with: i2cdetect -y 1.

Temperature + pH + TDS + LEDs

1. happyfishV4.py

   happyfishV5.py is a program designed to measure pH, TDS, and temperature sensor measurements. It also tracks LED brightness using LEDs and has a sunrise/sunset cycle. The script starts by importing necessary libraries, including ADS1115 and DFRobot_PH, which are used to interact with the pH and TDS sensor. The script defines two classes: Stages and Schedule. The Stages class defines the different stages of the day, including pre-sunrise, sunrise, daytime, sunset, and post-sunset. The Schedule class represents a schedule with sunrise and sunset times, and a duration for the sunrise and sunset periods. It provides methods to convert times to seconds, get the current stage based on the current time, and calculate the brightness percentage based on the current stage. The script sets up the PCA9685 PWM driver and defines a function set_pwm to set the PWM duty cycle for all channels based on the brightness percentage. The script creates ADS1115 objects for the pH and TDS sensors, sets their I2C addresses and gain values, and creates a DFRobot_PH object. In the main loop, the script continuously monitors and prints the real-time temperature, pH, and TDS values, as well as the current stage and brightness level. It also adjusts the LED brightness based on the current stage. The script can be stopped manually using Ctrl+C, and the PCA9685 PWM driver is cleaned up before exiting.

Hardware Requirements:

- Raspberry Pi

- Adafruit Cobbler (breakout board for Raspberry Pi)
- PCA9685 16-channel PWM driver
    - Connect VCC to Adafruit Cobbler's 3.3V pin
    - Connect GND to Adafruit Cobbler's GND pin
    - Connect SCL to Adafruit Cobbler's GPIO 5 (I2C clock)
    - Connect SDA to Adafruit Cobbler's GPIO 6 (I2C data)
- 16 LEDs connected to PCA9685 output channels (0-15)
    - Connect LED Positive leg to PCA9685 output channel (e.g., Channel 0)
    - Connect LED Negative leg to GND
- ADS1115 analog-to-digital converter
    - Connect VCC to Adafruit Cobbler's 3.3V pin
    - Connect GND to Adafruit Cobbler's GND pin
    - Connect SCL to Adafruit Cobbler's GPIO 5 (I2C clock)
    - Connect SDA to Adafruit Cobbler's GPIO 6 (I2C data)
- pH sensor
    - Connect to ADS1115 channel 2
- TDS sensor
    - Connect to ADS1115 channel 0
- DS18B20 temperature sensor
    - Connect VCC to Adafruit Cobbler's 3.3V pin
    - Connect GND to Adafruit Cobbler's GND pin
    - Connect DATA to Adafruit Cobbler's GPIO 4
- Breadboard and jumper wires for connections

Software Requirements:

- adafruit_pca9685 library installed
- CQRobot_ADS1115 and DFRobot_PH files in same directory

Steps to Run:

1. Connect the Raspberry Pi to the Adafruit Cobbler.
2. Connect the PCA9685 to the Adafruit Cobbler's I2C pins (SCL and SDA).
3. Connect the 16 LEDs to the PCA9685's output channels (0-15).
4. Connect the ADS1115 to the Adafruit Cobbler's I2C pins (SCL and SDA).
5. Connect the pH sensor to ADS1115 channel 2.
6. Connect the TDS sensor to ADS1115 channel 0.
7. Connect the DS18B20 temperature sensor to the Adafruit Cobbler's GPIO pins (e.g., GPIO 4).
8. Install the required libraries using pip: `sudo pip3 install adafruit-circuitpython-pca9685`
9. Run the script using Python: `python happyfishV5.py`
10. The script will continuously log the current time, stage, brightness level, pH, temperature, and TDS values.

Manual Commands:

- To stop the program, press `Ctrl+C` in the terminal.

Note: Make sure to adjust the `SUNRISE`, `SUNSET`, and `DURATION` environment variables to match your specific requirements.

Next Steps

1. **Integration with Web Portal**

- **Link Sensor Readings:** All data collected from the pH, temperature, and TDS sensors will be integrated into the project's dedicated web portal. This will allow for real-time monitoring and historical data analysis accessible from any device.
- **LED Light Stages Synchronization:** The various stages of the LED lighting, specifically those simulating sunrise and sunset, will also be linked to the web portal. This enables users to adjust lighting settings remotely and schedule automatic adjustments based on real-time environmental data.