CIWS-WM-Node Datalogger

Software Documentation

Software ver. 1.0.0

Project Organization

The CIWS-WM-Node software is comprised of the following files:

- LoggerAutoRun.py
- arduinoHandler.py
- piHandler.py
- LoggerShell_CLI.py
- logger.c
- setup.py
- setup.sh

The functions which interact with the Raspberry Pi GPIO peripherals are written in C code in logger.c. Using Python's API (available from Python.h), these C functions are wrapped up as a Python module called Logger, which is used in arduinoHandler.py. The software can be set up on the Raspberry Pi by running setup.sh, which configures the Raspberry Pi and calls setup.py, which compiles logger.c.

arduinoHandler.py Is a wrapper module for logger.c. It handles the communication of the pi with the microcontroller and EEPROM. It also handles the initialization of the SPI, UART, and GPIO pins. Pins 24 and 25 on the raspberry pi are set by this module to indicate that the EEPROM is being read and that the pi is on respectively.

LoggerAutoRun.py is run automatically whenever the Raspberry Pi is powered on. It is responsible for retrieving data from the EEPROM. It will also shut down the Raspberry Pi if the data buffer is full or it is midnight (in other words, if it was not turned on by a user).

piHandler.py handles the functionality of the pi. This includes the storage and retrieval of settings data. It also handles the analysis, storage, and transmission of both raw data and data resulting from the computations of raw data.

As the previous paragraphs suggest, with the exception of LoggerAutoRun.py, these scripts are run by LoggerShell_CLI.py. In fact, users should not run any other script than LoggerShell_CLI.py, which is run automatically when the user logs in. This script provides a user interface as a 'shell' around the functionality of the datalogger.

GPIO Functions: logger.c

The code in logger.c comprises the Logger module used in the other Python scripts. There are 9 functions in this file:

- init
- initPins
- bufferMax
- setRomBusy

- setPowerGood
- loadData
- reportSwap
- setRomFree
- setPowerOff

These functions mainly deal with GPIO and file I/O operations using the following libraries:

- wiringPi.h
- wiringPiSPI.h
- wiringSerial.h

The following sections describe the individual functions in this file.

Initializing the Module: init()

The function init initializes the wiringPi GPIO library, the wiringPi SPI library, and the wiringPi Serial library. The SPI module is initialized to communicate at 2 MHz, and the Serial module is initialized to communicate at 9600 Baud.

Initializing the GPIO pins: initPins()

This function sets up the initial values of the GPIO pins. There are two GPIO pins this function initializes:

- ROM_BUSY (Broadcom GPIO 24)
- POWER_GOOD (Broadcom GPIO 25)

ROM_BUSY and POWER_GOOD are both initialized as logic low (0 V).

Read Maximum Buffer Size: bufferMax()

This function simply returns the variable BUFFER_MAX. This value determines the amount of data that the Raspberry Pi reads from the EEPROM chip.

Request EEPROM: setRomBusy()

This function sets the ROM_BUSY pin HIGH, which tells the microcontroller that the Raspberry Pi is reading the EEPROM. The microcontroller yields the EEPROM in CERTAIN SITUATIONS:

1. The Raspberry Pi was just activated
2. The Raspberry Pi instructed the microcontroller to stop a logging session

The microcontroller will not honor this request in any other situations.

Report Power On: setPowerGood()

This function sets the POWER_GOOD pin HIGH, which tells the microcontroller that the Raspberry Pi has successfully powered on.

Read Data from EEPROM: loadData()

This function reads data from the EEPROM chip. The function allocates an array to hold the EEPROM data. The function then calls a SPI transaction, and data gets transferred from the EEPROM chip to the data array.

Swap Report with Microcontroller: reportSwap()

This function swaps "report" data with the Arduino. This data consists of the following:

Data Field (Bytes)	Name	Description
0	Years	Current year
1	Months	Current month
2	Days	Current day
3	Hours	Current hour
4	Minutes	Current minute
5	Seconds	Current second
6	Previous Pulses	Pulses in the previous logging sample period
7	Total Pulses 0	Total pulses over logging session (byte 0)
8	Total Pulses 1	Total pulses over logging session (byte 1)
9	Total Pulses 2	Total pulses over logging session (byte 2)
10	Logging	Boolean: Is the device logging or not?

The data swap is done over a serial UART bus. The data messages are interlaced; in other words, the Raspberry Pi sends one byte, and the Arduino sends the corresponding byte. The Raspberry Pi can overwrite the data by sending any data value other than 0xFF, or 255. 255 (0xFF) is used by the Raspberry Pi to read the data.

Release EEPROM: setRomFree()

This function simply sets the ROM_BUSY pin low, signaling to the Arduino that the Raspberry Pi has finished reading the EEPROM.

Report Power Off: setPowerOff()

This function sets the POWER_GOOD pin low, signaling to the Arduino that the Raspberry Pi is shutting down. The Arduino will cut power to the Raspberry Pi 12 seconds after receiving this information.