CIWS-WM-Node

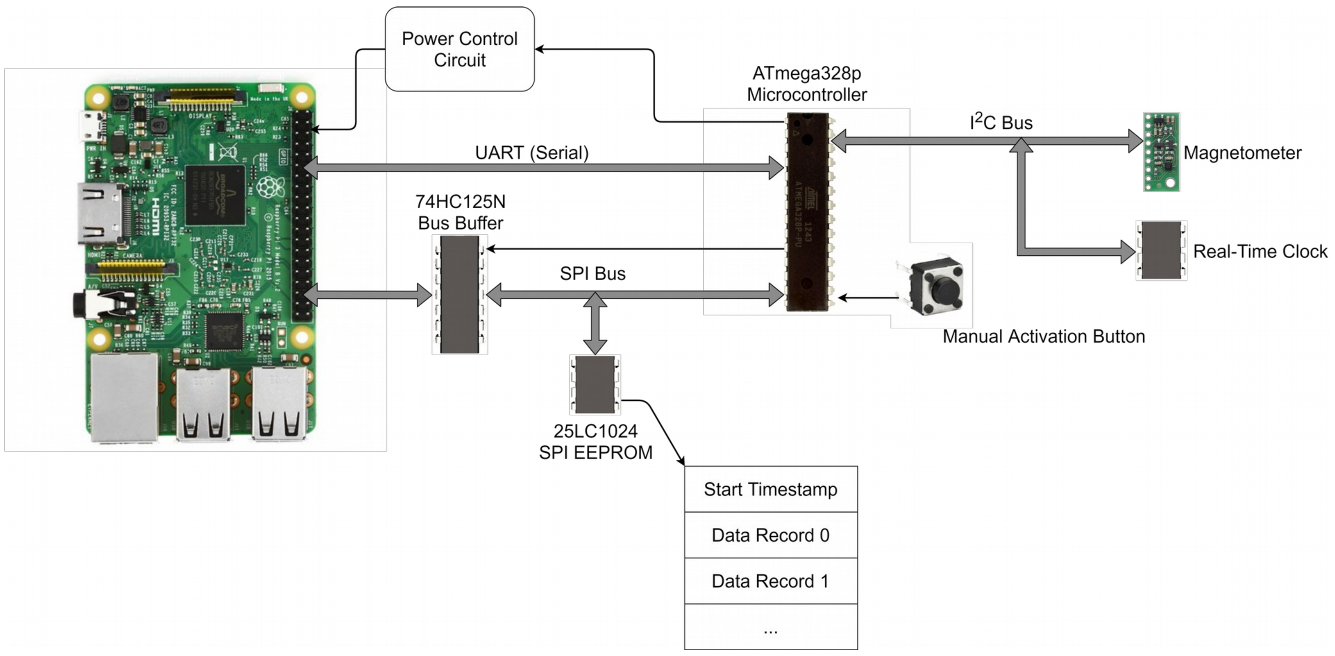
Hardware Documentation

Overview

The WM-Node datalogger consists of the following main components:

1. AVR microcontroller
2. Raspberry Pi embedded computer
3. EEPROM flash storage
4. Bus buffer
5. Power control circuit
6. Real-Time Clock
7. Magnetometer
8. Manual activation button

A diagram of the WM-Node architecture is shown here:



The more critical components will be described in detail throughout this document. In brief, the microcontroller collects pulse data and stores it on the SPI EEPROM. The Raspberry Pi, when powered by the microcontroller through the power control circuit, reads the pulse data from the EEPROM, processes the data, and stores the data in a file.

The microcontroller and the Raspberry Pi communicate with each other using a UART (serial) connection, not over the SPI bus, which is only used for the EEPROM in this design. Currently, the communication protocol over UART is simple, but may be prone to failure. A more robust communication protocol over UART is being developed.

AVR Microcontroller

The microcontroller, an ATmega328p from Microchip/Atmel, is an 8-bit AVR core microcontroller, known especially for its use in many Arduino platforms; in fact, a 3.3 V 8 MHz Arduino Pro was used for the prototype.

The microcontroller is primarily responsible for the following tasks:

1. Collect raw data from the magnetometer
2. Detect pulses in the data
3. Store pulses to the EEPROM
4. Control power to the Raspberry Pi
5. Control the Raspberry Pi's access to the EEPROM
6. Communicate with the Raspberry Pi
    - This includes commands to start/stop logging from the user, water flow data, date/time, etc.
7. Generate timestamps using information from the RTC.

Information on the firmware for this device is coming soon.

Raspberry Pi Embedded Computer

The Raspberry Pi is responsible for processing the data collected by the microcontroller, including disagreggation and storage. The Raspberry Pi will also be responsible for transmitting the data over the internet (via WiFi, cellular, or some other means) to a server.

On top of this technical functionality, the Raspberry Pi also implements a user interface for the datalogger (see the Software Documentation). Settings set by the user are communicated to the microcontroller, and information from the microcontroller is communicated from the microcontroller to the Raspberry Pi and, and then to the user.

EEPROM Flash Storage

The 25LC1024 SPI EEPROM chip can store 128kB of data from the microcontroller. The data stored in the chip is in the following format:

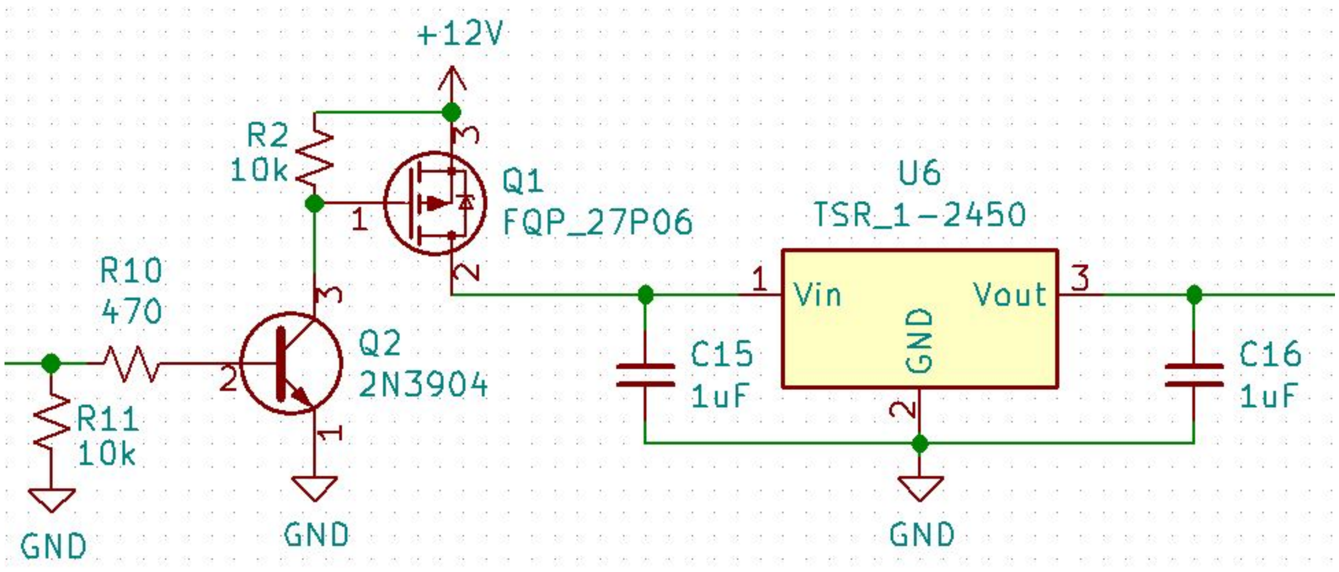
Bytes 0-2	Number of data records
Bytes 3-8	Starting timestamp
Bytes 9-N	Data bytes

Bus Buffer

The bus buffer is implemented using a 74HC125N buffer chip. The SPI bus on the Raspberry Pi connects to this chip. The chip also connects to the microcontroller's SPI bus, which connects in turn to the EEPROM chip. This buffer is controlled by the microcontroller. The end result is that when the buffer is activated by the microcontroller, the Raspberry Pi is connected to the EEPROM chip. When the buffer is deactivated by the microcontroller, the Raspberry Pi is disconnected from the EEPROM chip. This permits the SPI bus to be used while the Raspberry Pi is off (driving the I/O pins of the Raspberry Pi while it is powered off can cause damage to the Raspberry Pi).

Power Control Circuit

The power control circuit switches power on and off to the Raspberry Pi, and is controlled by the microcontroller. The following diagram is from the WM-Node schematic:



This looks a bit complicated, but don't worry, it's actually quite simple. Start with the voltage regulator, U6. This component converts the battery voltage from 12 volts down to 5 volts for the Raspberry Pi. The two capacitors, C15 and C16, simply smooth out the power supply voltage.

Notice that the transistor Q1 sits between the battery (+12V) and U6. Q1 acts like a switch. When it's turned on, U6 will output 5 volts. When Q1 is off, U6 will have no input voltage, and will therefore give no output voltage.

To turn Q1 on, the 'gate' pin (labeled '1') must be set to 12 volts. To turn Q1 off, the 'gate' pin must be set to 0 volts. The microcontroller can set the pin to 0 volts with no issues; however, the microcontroller by itself can only set a pin to 3.3 volts because its supply voltage is 3.3 volts.

To remedy this problem, a second transistor (Q2) is used. Q2 is a different kind of transistor, and works differently than Q1. To turn on Q2, electric current must flow into it through pin 2, the ‘base’ pin. To turn off Q2, no current can flow. This is something the microcontroller can control.

When Q2 is ‘on’, the gate pin on Q1 is connected to zero volts, which ensures that Q1 is ‘off’ and cannot conduct. This then turns off the voltage regulator U6, which turns off the Raspberry Pi. When Q2 is ‘off’, the gate pin of Q1 is no longer connected to zero volts; instead, it is connected to 12 volts through the resistor R2. This allows Q1 to conduct. This turns on the voltage regulator U6, which turns on the Raspberry Pi.

Magnetometer

The magnetometer used in this prototype is an LIS3MDL magnetometer by ST Microelectronics. The LIS3MDL we use is mounted to a board and sold by Pololu.

The LIS3MDL magnetometer has several configurable sample rates. The sample rate used in this project is 560 Hz (Camilo’s first paper discusses this sampling rate in further depth). When the LIS3MDL signals that data is ready, the data is read by the microcontroller using the I^2C serial bus, and is processed and stored on the EEPROM chip.

Real-Time Clock

The Real-Time Clock (RTC) used in this prototype is a PCF8523 by NXP Semiconductor. This RTC is present on the Arduino Datalogging shield, and was thus a simple prototyping choice for the RTC. The PCF8523 also communicates with the microcontroller using I^2C, and shares the I^2C bus with the LIS3MDL magnetometer.

The RTC is used by the microcontroller for sample interval timing and timestamp generation. The RTC signals the microcontroller whenever a specified interval has passed, and the microcontroller then counts up the detected pulses and stores that value. The date/time is also read from the RTC when this interval has passed, from which the microcontroller creates a timestamp.

Manual Activation Button

This button is used to start up the Raspberry Pi manually. This is useful because the Raspberry Pi is turned on and back off about once per day, and it is inconvenient for the user to wait until the Pi turns on to do anything with the Pi. When the button is pressed, the Pi powers on and, instead of automatically turning back off, it will wait for the user to log in. The user can then log in to the system and view files, start/stop a logging session, view water flow data, and other various tasks.