

A Raspberry Pi add-on

DIY a water detection from a small cpu

Daniel Perron
20 november 2013

Preface

This is about a way to detect water inside a pipe without any contact with the water itself. The best approach found, in my perspective is to utilise the dielectric properties between the water and the air. If you add water between a capacitor plate the capacitance will increase.

The PIC12F1840 has capacitive sense capabilities. The system is a very weak up/down oscillator. If one of the pin is connected to a bigger capacitor, the frequency changes and using a time event we could calculate the actual frequency at this way the value of the capacitance.

Using two metal plate on each side of a pipe, we are creating a capacitor, by filling the tube with water, we are changing the electrolytic value and then the frequency of the system.

Since we want to implement multiple sensors and minimize the number of wires, the RS-485 was the best choice. Simple enough to transport signal with minimal noise and having a way to switch between who is talking and receiving.

I choose Modbus for the communication protocol. Don't need to re-invent. This is very popular and simple to implement. A good reason to use Modbus was the existence of module already available in python to deal with Modbus. The python module is `minimalmodbus`.

And finally The Raspberry Pi need a way to communicate with the RS-485. Since I decide the half-duplex mode, Only one talk at a time, I need to create an automatic switch system that will flip the RS-485 signal from receive mode to transmit mode. I did add another cpu which deal with the communication direction.

Two different modules to build. The first one is the water detection module and the second one is the Rs-485 interface to the Raspberry Pi..

This project is for fun and I think it is a good project to increase electronic and programming skill. I will try to make it as simple it could be.

Daniel Perron

Step 1 - Getting what you need.

You have a Raspberry Pi and you want to create module for water detection .
You will need some tools and parts. So let's make a list.

- Raspberry Pi (model A or B).
- Power supply for the Raspberry Pi.
- Power supply for all the modules. Best choice will be 9V 300ma.
- Protoboard to test and mount design. Better to test before soldering everything.
- Wire kits to easily plug components together.
- Electronic pliers and cutter.
- Soldering Iron and electronic soldering wire. (sponge to clean soldering head).
- Electric tape to prevent short cut. I use it at the end of barenaked power supply wire.

And now the parts

The Water Detect Module,

- Veroboard with 0.1" hole spacing. This is the final board which everything will be in.
- PIC12F1840 8 pin DIP Microchip cpu.
- LTC485 RS-485 8 pin dip. This one is from Linear technology but you could use similar device from Maxim or Ti.
- 3 X 0.1 μ F 25V ceramic capacitor.
- 10 μ F 25V electrolytic capacitor.
- 1 μ F 10V electrolytic capacitor.
- A screwable 4 Pins connector. (0.2" spacing is very good to solder on veroboard).
- LM7805 style 5V regulator. TO92 case will do put a TO220 is ok.
- One Led to display signal if you want
- And one 220 Ω 1/4 W resistor
- Plastic box to put everything in. Maybe and electric box for outside will be ok.
- Aluminium tape. This will be the capacitor plate glue to the pipe.

The RS-485 Switch interface,

- Veroboard with 0.1" hole spacing.
- PIC12F1840 8 pins DIP Microchip cpu.
- LTC485 RS-485 8 pin dip. or RS485 driver from maxim or TI.
- 3 X 1K Ω resistor $\frac{1}{4}$ W.
- 4K7 Ω resistor $\frac{1}{4}$ W.
- 10K Ω resistor $\frac{1}{4}$ W.
- 120 Ω resistor $\frac{1}{4}$ W.
- 2 X 0.1 μ F 25V ceramic capacitor.
- A screwable 4 Pins connector. (0.2" spacing is very good to solder on veroboard).
- A screwable 2 Pins connector. (0.2" spacing is very good to solder on veroboard).
- A 26 Pin header or some 4 pins header to pass the RPi signal to the interface

Plus,

- Twist pair cable , 2 pairs to connect the two modules together. Network cable will Do. I use CAT3 2 pairs, awg 24.
- 120 Ω resistor $\frac{1}{4}$ W. This is use at the end of the cable . (Terminator)

Step 2 - Setup the Raspberry Pi

We are using the internal serial communication of the Raspberry Pi. We will need to disable the debug and tty console on the raspbian OS.

With Jessie and Wheezy it is now possible to disable the debug and the tty console with the raspi-config application.

Please use

- Execute **sudo raspi-config**.
- Go to the **Advanced Option**.
- Choose **A8 Serial**.
- Select **<No>**.

This is the old method. You could skip 1 and 2.

1 - Modification of `/boot/cmdline.txt` to disable the `/dev/ttyAMA0`

- Change directory to `/boot`.
`pi@raspberrypi ~ $ cd /boot`
- Make a backup copy just in case.
`pi@raspberrypi /boot $ cp cmdline.txt cmdline.txt.bk.1`
- Edit `cmdline.txt` and remove anything related to `/dev/ttyAMA0`.
`pi@raspberrypi /boot $ sudo nano cmdline.txt`
- remove **console=ttyAMA0,115200 kgdboc=ttyAMA0,115200**
- **ctrl-O** to overwrite and **ctrl-X** to exit.

2 - Removal of `/dev/ttyAMA0` console in `/etc/inittab`.

- Edit the file `/etc/inittab`.
`pi@raspberrypi /boot $ cd`
`pi@raspberrypi ~ $ sudo nano /etc/inittab`
- locate the line with `/dev/ttyAMA0` (The last one)
- Add a `#` at the beginning to disable it
`#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100`
- **ctrl-O** to overwrite and **ctrl-X** to exit.
- Reset `inittab` with the `init q` command
`pi@raspberrypi ~ $ sudo init q`

3 - Installation of minimalmodbus python module.

- Let's update the RPi first.

```
pi@raspberrypi ~ $ sudo apt-get update
```

- Install python-pip.

```
pi@raspberrypi ~ $ sudo apt-get install python-pip
```

- Install minimalmodbus.

```
pi@raspberrypi ~ $ sudo pip install -U minimalmodbus
```

4 - Installation of intelhex python module.

- Get the source code.

```
pi@raspberrypi ~ $ wget http://www.bialix.com/intelhex/intelhex-1.5.zip
```

- Unzip it.

```
pi@raspberrypi ~ $ unzip intelhex-1.5.zip
```

- Install the module.

```
pi@raspberrypi ~ $ cd intelhex-1.5/
```

```
pi@raspberrypi ~/intelhex-1.5 $ sudo python setup.py install
```

5 - Installation of git.

- Install github.

```
pi@raspberrypi ~/intelhex-1.5 $ cd
```

```
pi@raspberrypi ~ $ sudo apt-get install git
```

6 - Personal modbus folder download.

- This is the simplest method. Just download everything on github.

```
pi@raspberrypi ~ $ git clone https://github.com/danjperron/NoContactWaterDetect
```

7- BurnLVP. python code to program PIC cpu.

- Download from github.

```
git clone https://github.com/danjperron/burnLVP
```

8- Reboot

```
sudo reboot
```

Step 3 - Program The PIC cpu

The PIC12F1840 had 4K words eeprom to hold to program. It is possible to program the cpu using low voltage programming mode (LVP). We will use the Raspberry Pi to insert the binary code into the PIC. The burnLVP python code convert the Raspberry Pi into a Microchip programmer in LVP mode.

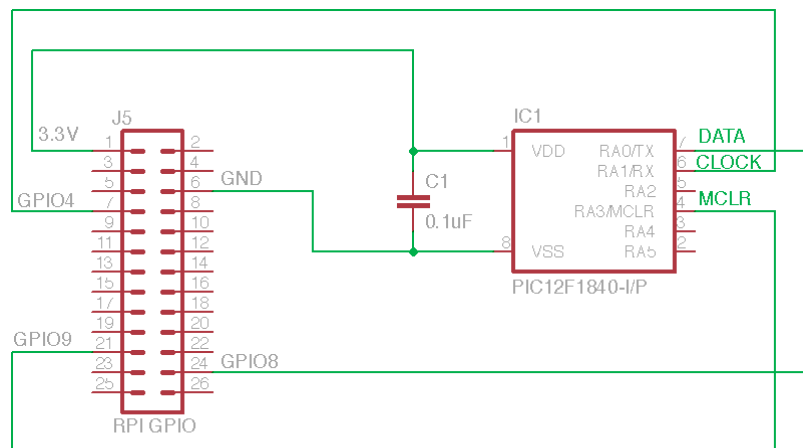
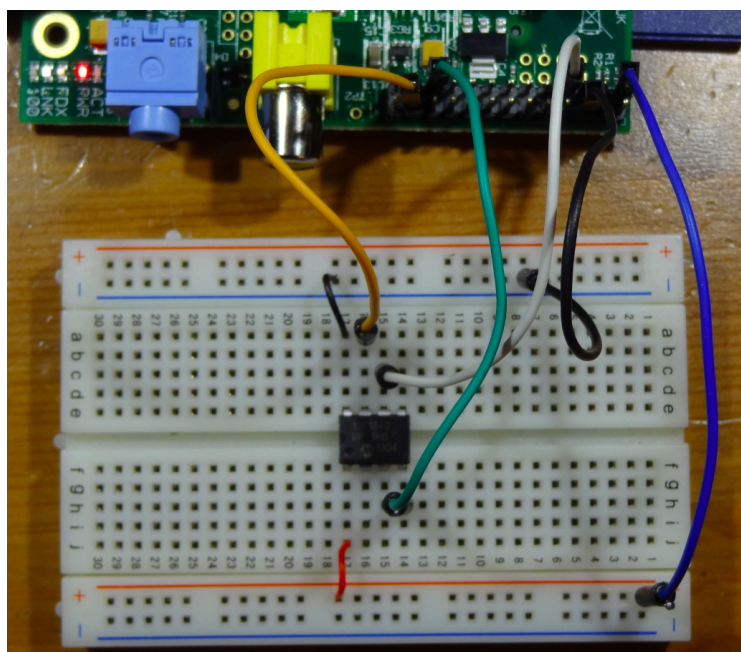


Figure 1. Raspberry Pi programmer for Microchip PIC cpu.



Picture 1. PIC12F1840 wired to be programmed.

1 - Program the PIC12F1840 for the Water Detect module.

- Insert the CPU PIC12F1840 into a protoboard and connect wires according to Figure 1.
- Run the burnLVP python application with the file argument WaterDetectModbus.hex

```
pi@raspberrypi ~ $ sudo ./burnLVP/burnLVP.py ./NoContactWaterDetect/WaterDetectModbus.hex
```

The application should output

```
File " ./NoContactWaterDetect/WaterDetectModbus.hex " loaded
LVP ON
Cpu : 0x1b80 : PIC12F1840 Revision 0x4
ProgramSize = 0x1000
DataSize   = 0x100
Bulk Erase Program , Data. .... done.
Program blank check.....Passed!
Data Blank check.....Passed!
Writing Program.....Done.
Program check .....Passed!
Writing Data.Done.
Data check .Passed!
Writing Config.....Done.
Config Check.....Passed!
No Error. All Done!
LVP OFF
```

- Unplug 3.3V power wire and remove cpu from protoboard
- Use Label sticker and mark the chip . I use “WD”

2 - Program The RS-485 cpu the same way but use the file rs485Switch.hex

```
pi@raspberrypi ~ $ sudo ./burnLVP/burnLVP.py ./NoContactWaterDetect/rs485switch.hex
```

Step 4 - Cpu verification

The water detect module cpu has been programmed. It should be able to interpret Modbus command. We Should check the programming of by running a Modbus command. We don't need to use rs-485 interface yet since the Rpi and the cpu could talk directly using the asynchronous port of each other. The rs-485 is simply a physical transport.

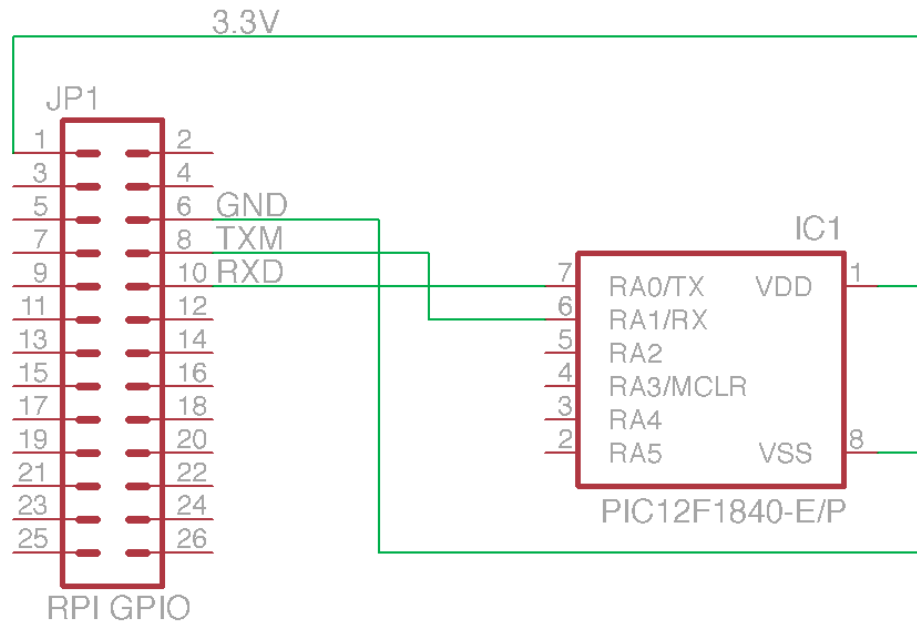


Figure 2. Water Detect module cpu in test mode.

1 - Using the python application "TestSlave127.py", the application should respond.

```
pi@raspberrypi ~ $ cd NoContactWaterDetect/
```

```
pi@raspberrypi ~/NoContactWaterDetect $ ./TestSlave127.py
```

```
Callbration for Air Frequency : 3400
```

```
Calibration for Water Frequency : 2400
```

```
F: 15628
```

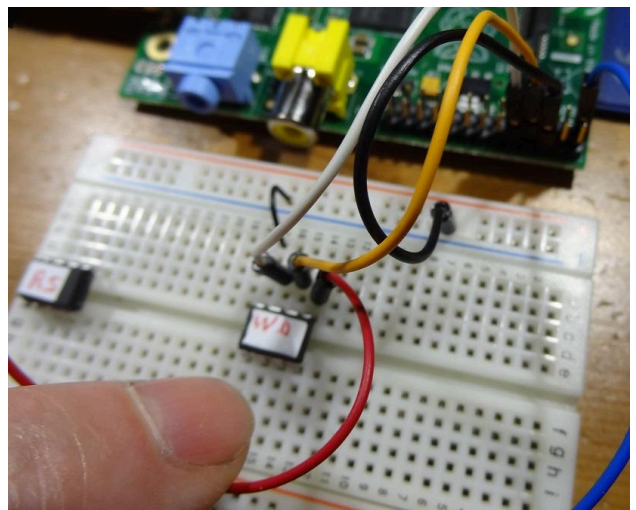
```
F: 15633
```

```
F: 15630
```

```
F: 13048
```

```
F: 12112 <- wire been touched
```

```
F: 12243
```



Step 5 - Slave Address

After the pic cpu is programmed, it has the modbus slave address 127. Since we want more than 1 module we will need to change the address. A simple write from modbus function 6 into address 2 will change the corresponding Slave address of the module. I made a small python program called "SlaveAddress.py". I put the debug mode on it to show you what happens when you change the address.

```
pi@raspberrypi ~/NoContactWaterDetect $ ./SlaveAddress.py 127 1
Current Address is 127
New Address will be 1
MinimalModbus debug mode. Writing to instrument: '\x7f\x06\x00\x02\x00\x01\xe3\xd4'
MinimalModbus debug mode. Response from instrument:
'\x01\x06\x02\x00\x02\x00\x01\x12f'
Traceback (most recent call last):
  File "./SlaveAddress.py", line 23, in <module>
    instrument.write_register(2,NewAddress,0,6);
  File "/usr/local/lib/python2.7/dist-packages/minimalmodbus.py", line 247, in
write_register
    self._genericCommand(functioncode, registeraddress, value, numberOfDecimals,
signed=signed)
  File "/usr/local/lib/python2.7/dist-packages/minimalmodbus.py", line 646, in
_genericCommand
    payloadFromSlave = self._performCommand(functioncode, payloadToSlave)
  File "/usr/local/lib/python2.7/dist-packages/minimalmodbus.py", line 727, in
_performCommand
    payloadFromSlave = _extractPayload(response, self.address, functioncode)
  File "/usr/local/lib/python2.7/dist-packages/minimalmodbus.py", line 887, in
_extractPayload
    responseaddress, slaveaddress, response))
ValueError: Wrong return slave address: 1 instead of 127. The response is:
'\x01\x06\x02\x00\x02\x00\x01\x12f'
pi@raspberrypi ~/NoContactWaterDetect $
```

The python application tell us that there is an error. The module with address 127 return the wrong slave address and it is 1. it means that the module is now at the correct slave address . Maybe I will change the code not to give this error.

Step 6 - Prototype assembly

We would put the full assembly layout on a protoboard. It is always easier to fix stuff by moving wires or swapping parts without using a soldering iron. The protoboard schema will be first without an external power supply.

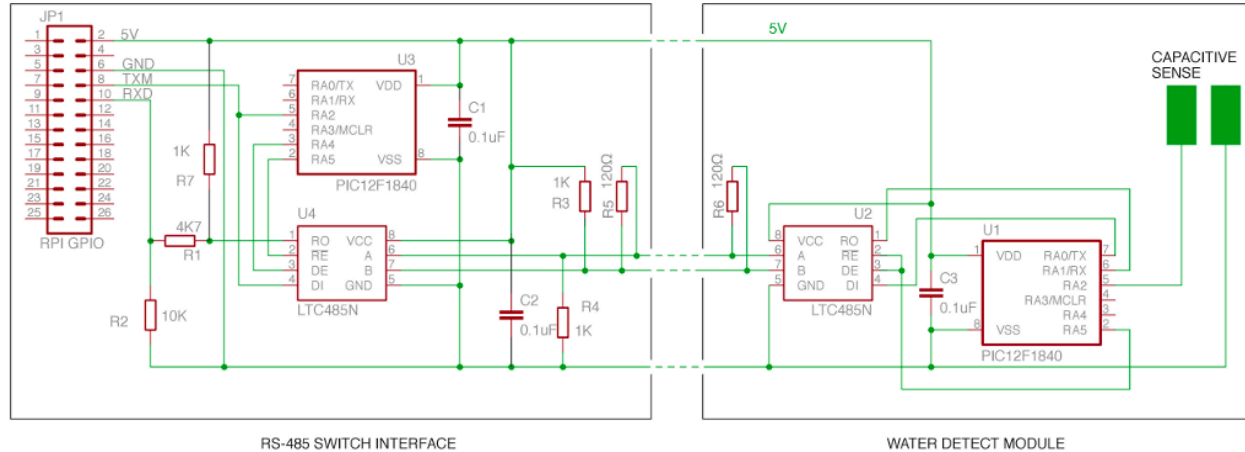
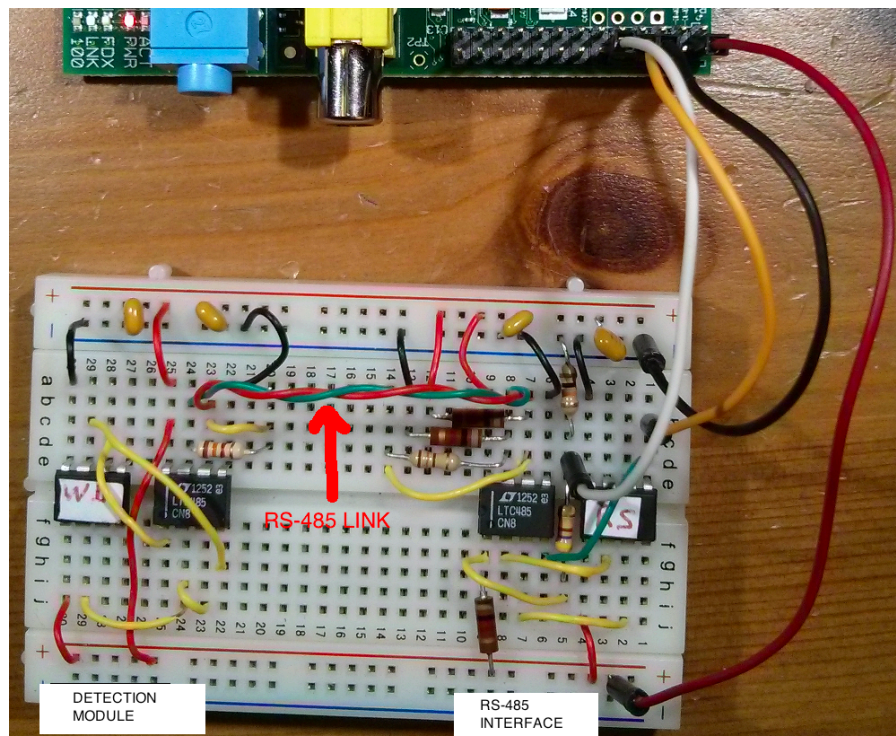


Figure 3 - Protoboard Water Detect assembly

When everything is assembled, the CheckModbus.py python routine should work. Don't forget to change the slave address in the python code since it is set to 1. The next step will be to add the Voltage regulator on the Water Detect Module. to finalize the Water Detect module schematic.



Picture 3. Module and RS-485 interface

Step 7 - Detector assembly

Cut the veroboard to fit in the box. Be sure that all components will fit on it. I use IC socket. This way it is easier to reprogram the cpu. Schema on Figure 5 (Appendix).

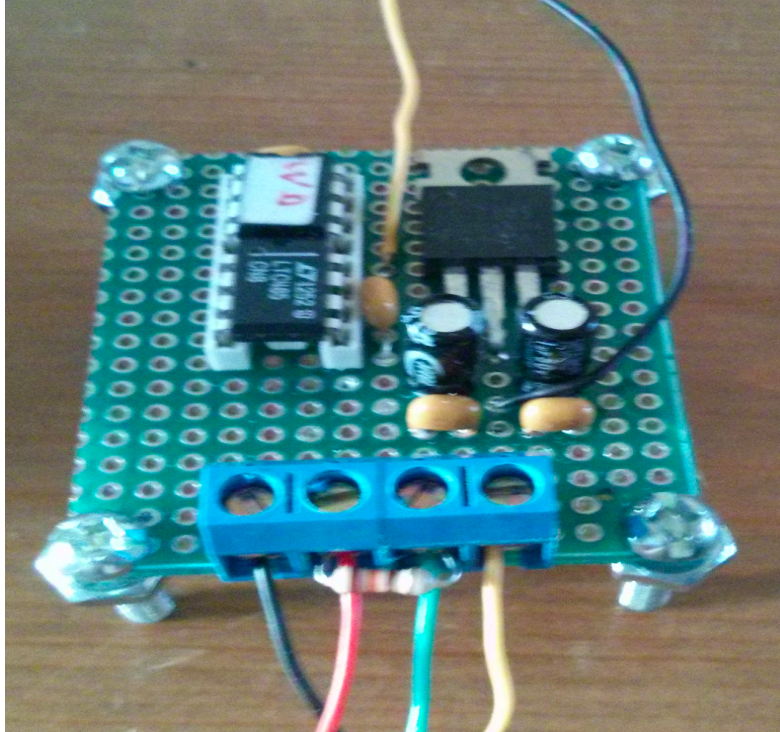


Photo 4 - The detector board.



Photo 5- The detector in the box.

The real detector is the aluminium tape on each side of the 21mm tube. The tape acts like a capacitor. Remove isolation on the wire and create a zigzag pattern to increase contact area. Use a second aluminium tape to secure the wire.



Photo 6 - ZigZag wire to increase contact area.

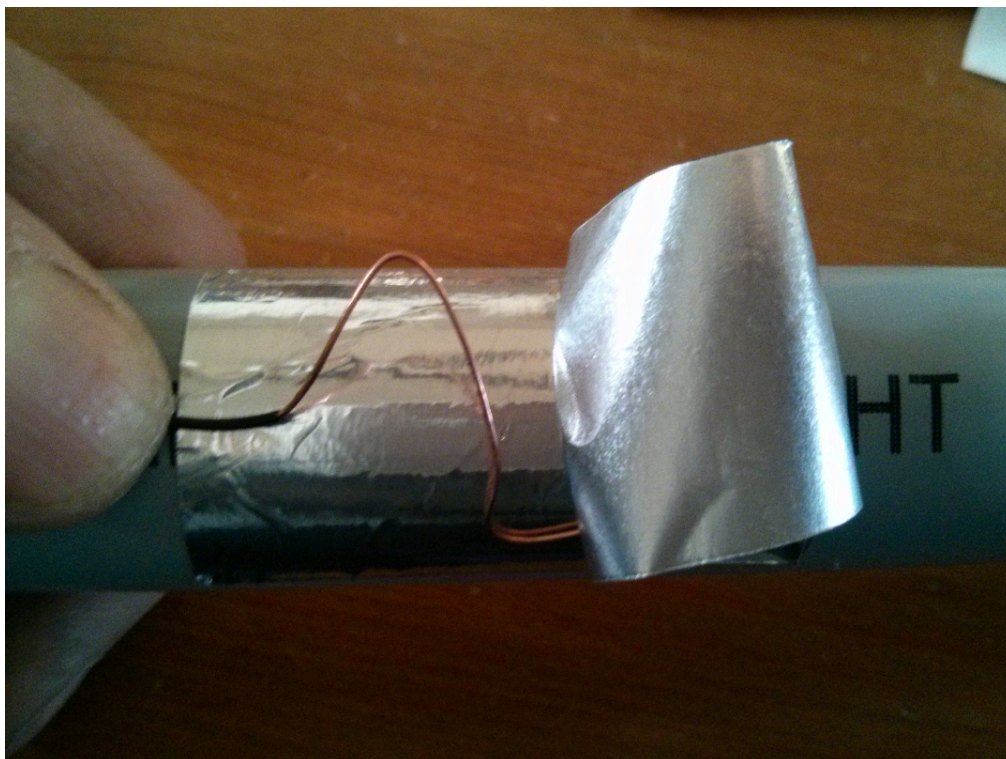


Photo 7 - Use an other piece of aluminium tape.



Photo 8- Everything together except a bracket to hold the box on the pipe.

You should use a heat shrink tube with silicon at each end to seal it.

Step 8 - Web page interface

I use webiopi ! this is a python web server. You could add reference function into it. I just add the read and write modbus function.

Follow the instruction from <https://code.google.com/p/webiopi/wiki/INSTALL> to install webiopi.

This is basically,

```
$ wget http://webiopi.googlecode.com/files/WebIOPi-0.6.0.tar.gz
$ tar xvfz WebIOPi-0.6.0.tar.gz
$ cd WebIOPi-0.6.0
$ sudo ./setup.sh
```

Please change index.html and ModbusSettings.html to fit your needs.

To start the application just run

```
cd
cd NoContactWaterDetect
sudo python WebModbus.py
```

And now just open your web browser and go to the url

`http://<you raspberry pi IP>:8000`

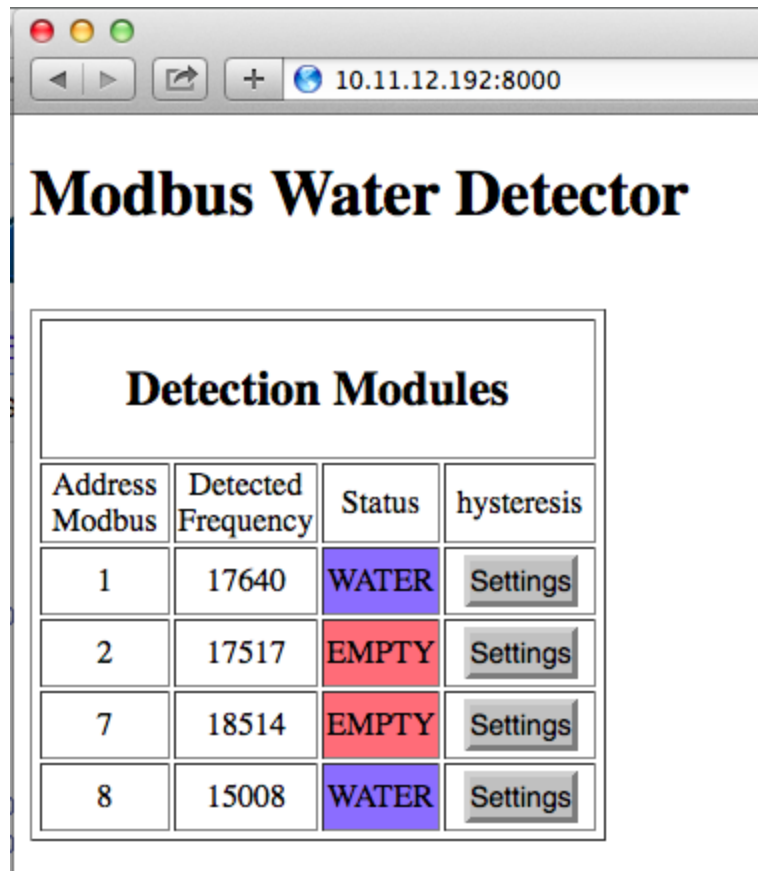
P.S. You could find the Raspberry Pi with the command

```
sudo ifconfig
```

check for inet addr: string in eth0 or wlan0 if you have wifi.

The index.html web page use javascript and provide a way to select the valid modbus modules. Check in the html javascript code the variable called 'ValidModuleList'. You should specify all the water detection module you have online.

This is a sample of the Webpage output



Detection Modules			
Address Modbus	Detected Frequency	Status	hysteresis
1	17640	WATER	Settings
2	17517	EMPTY	Settings
7	18514	EMPTY	Settings
8	15008	WATER	Settings

Figure 4 - Web page interface with 4 modules

Conclusion

I will let you do the RS-485 interface on veroboard. The schema is on figure 6 in the appendix.

this is it for me.

Now you should be able to add anything you want on Modbus, I/O, A/D , Frequency counter, etc... You need a knowledge in C but you have an example.

Have Fun,

Daniel Perron

Reference

Modbus Protocol Reference Guide
Modicom PI-MBUS-300 Rev. H
AEG SCHNEIDER
AUTOMATION

PIC12F1840
Microchip
8-Pin Flash Microcontrollers with XLP Technology
<http://ww1.microchip.com/downloads/en/DeviceDoc/40001441D.pdf>

LTC485
Linear Technology
Low power RS485 Interface transceiver
<http://cds.linear.com/docs/en/datasheet/485fi.pdf>

APPENDIX

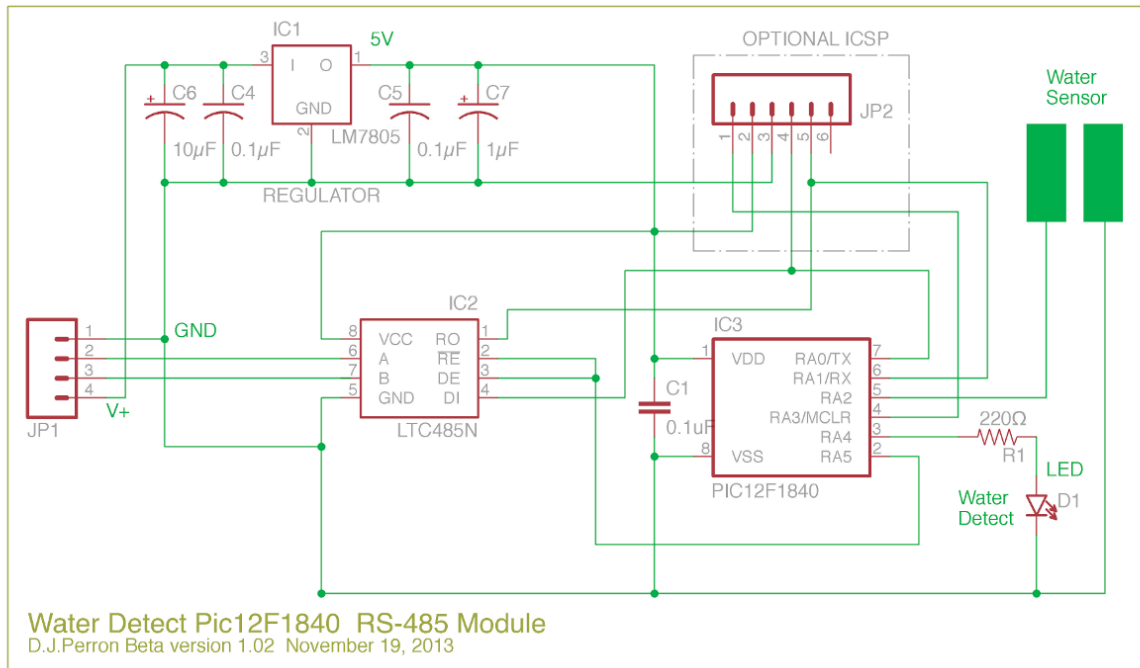


Figure 5 - Water Detect module schematic.

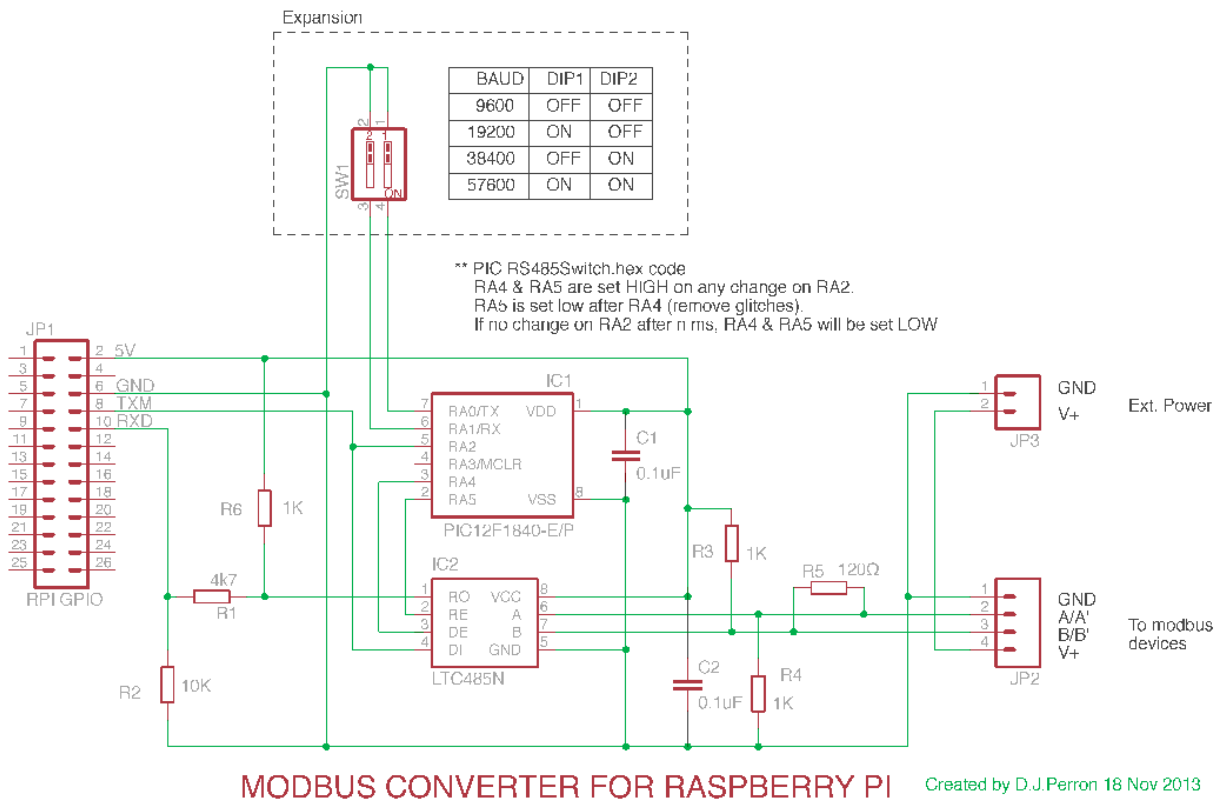


Figure 6- RS-485 Switch interface

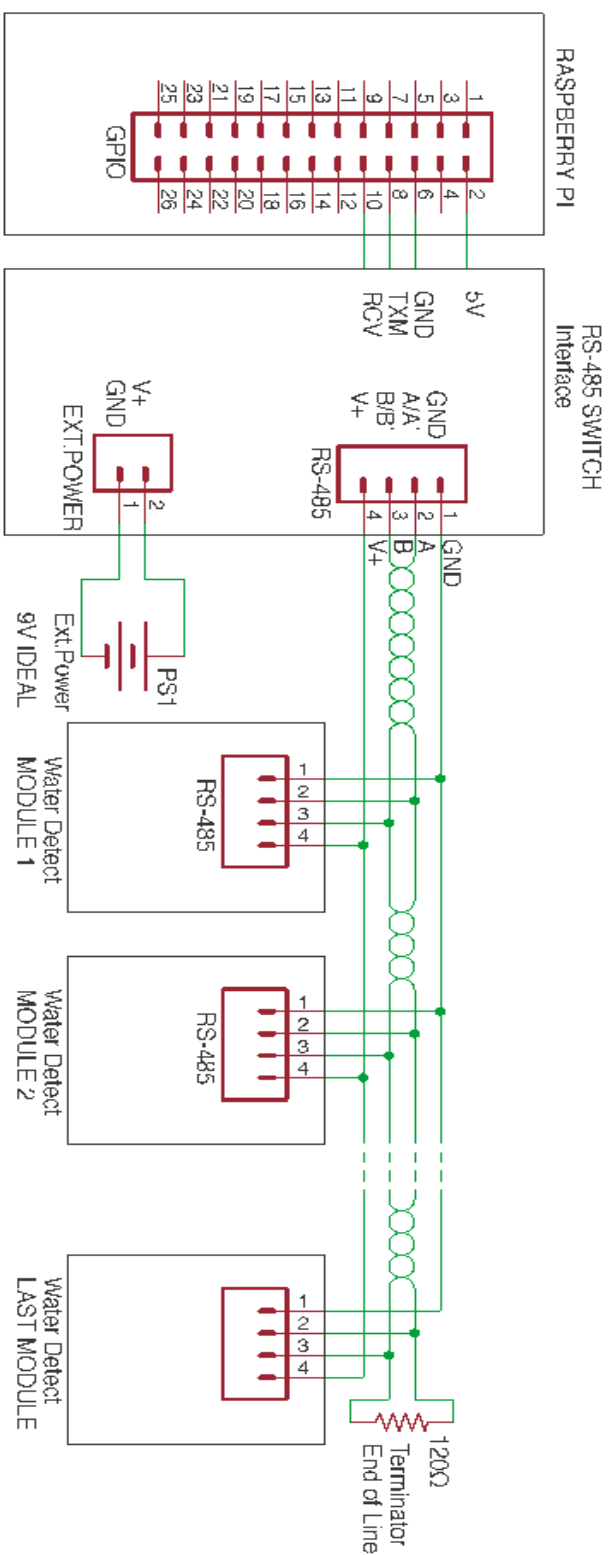


Figure 7 - System Layout.

Raspberry Pi Modbus layout with Water Detect modules.

D.J.Perron 19 November 2013