Modbus sur le Raspberry Pi

Faites le vous même
Détecteur de liquide avec une puce PIC

Daniel Perron
20 novembre 2013

Préface

Après avoir discuté sur le forum de RaspberryPi.org. Il m'est venu une idée pour détecter si il y a de l'eau dans un tuyau de PVC. J'ai donc utilisé une puce Microchip qui possède une périphérique de captation capacitive. C'est un oscillateur donc la patte du micro-processeur varie la fréquence par ajout d'une capacité parasite qui normalement serait notre doigt. Avec deux feuilles d'aluminiums de chaque côté du tuyau nous formons un condensateur. Lorsque l'eau rempli l'espace dans le tuyau , nous changeons le matériel donc la constance diélectrique. Et de ce fait la fréquence de l'oscillateur diminue.

En utilisant Modbus, nous pouvons brancher plus d'un module sur le Raspberry Pi. et de ce fait ajouter d'autre périphérique. Minimalmodbus avec python donne un façon facile de communiquer avec les modules.

Le port série interne du RPi est de 3.3 V. L'utilisation de la couche RS-485 va permettre d'allonger la distance possible des modules, d'isoler le RPi et de brancher tout les modules en mode maître/esclave. Ce qui veut dire que le RPi est le maître et que les modules sont esclaves. Le maître parle et les modules répondent.

J'utilise seulement le RPi avec la réception et transmission sérielle. J'ai donc besoin d'un autre PIC cpu pour la commutation de la direction du signal.

J'ai fait ce projet pour le plaisir. Il est donc possible qu'il y a des erreurs et qu'il a besoin de plus de travail pour qu'il soit vraiment au point.

Daniel Perron

Étape 1 - Le matériel.

Vous avez un Raspberry Pi , c'est un début. Voici une liste des choses que voud devez avoir.

- Raspberry Pi (modèle A or B).
- Adaptateur de tension pour le RPi.
- Adaptateur externe pour les modules. Une tension de 9V serait idéale.
- Une planche de test type "protoboard" pour tester le désign.
- Un kit de cable avec pin et connecteur simple pour facilement brancher le tout.
- Des outils! Pinces pour couper et plier.
- Un fer à souder avec de la soudure.
- Du ruban électrique pour isoler les fils si nécessaire.

et Maintenant les composantes électronique.

Le module de détection ,

- Plaque électronique type "véroboard" à espacement des trous de 0,1". C'est pour le montage final.
- PIC12F1840 8 pin DIP Microchip cpu.
- LTC485 RS-485 8 pin dip. Puce de Linear technology mais vous pouvez utiliser Maxim ou TI, etc..
- 3 X 0.1μF 25V condensateur céramique.
- 10 μF 25V condensateur électolytique.
- 1 μF 10V condensateur électolytique.
- Connecteur à vis 4 broches pour brancher les fils electrique. Espacement 0,2"
- Régulateur de tension type 7805. boîtier TO-92 ou TO-220.
- D.E.L. pour afficher l'alarme.
- Résistance 220Ω ¼ W.
- Boîter de plastique pour mettre le tout. (Boîte électrique serait idéal).
- Ruban Allumisé. Le ruban fera les plaques pour le condensateur de détection.

L’interface RS-485,

- Plaque électronique type “véroboard” à espacement des trous de 0,1”.
- PIC12F1840 8 pins DIP Microchip cpu.
- LTC485 RS-485 8 pin dip. Puce de Linear technology mais vous pouvez utiliser Maxim ou TI, etc..
- 3 x résistance 1KΩ ¼ W.
- Résistance 4K7Ω ¼ W.
- Résistance 10KΩ ¼ W.
- Résistance 120Ω ¼ W.
- 2 X condensateur céramique 0.1μF 25V.
- Connecteur à vis 4 broches pour brancher les fils électriques. Espacement 0,2”
- Connecteur à vis 2 broches pour brancher les fils électriques. Espacement 0,2”
- Connecteur femelle 26 broches pour brancher sur le GPIO du RPi.

Plus,

- Cable avec 2 paires torsadées. J’utilise du CAT 3 , 2 paires , AWG24.
- Résistance 120Ω 1/4W. Résistance de terminaison pour fin de câble.

Étape 2 - Préparation du Raspberry Pi

Nous utilisons la périphérique de communication série interne du Raspberry Pi. Nous devons désactiver la console de débugage et la console de login série.

1 - Modification de /boot/cmdline.txt to désactiver /dev/ttyAMA0

- Changer le répertoire pour /boot.
  pi@raspberrypi ~ $ cd /boot

- Faire un copie de sauvegarde au cas ou.
  pi@raspberrypi /boot $ cp cmdline.txt cmdline.txt.bk.1

- Modifier cmdline.txt and et supprimer tout ce qui est /dev/ttyAMA0.
  pi@raspberrypi /boot $ sudo nano cmdline.txt

  enlever **console=ttyAMA0,115200 kgdboc=ttyAMA0,115200**

  ctrl-O pour enregistrer et **ctrl-X** pour sortir.

2 - Supprimer /dev/ttyAMA0 dans /etc/inittab.

- Modifier /etc/inittab.
  pi@raspberrypi /boot $ cd
  pi@raspberrypi ~ $ sudo nano /etc/inittab

- Trouver la ligne avec /dev/ttyAMA0 (La dernière)

- Ajouter '#' au début de la ligne pour la désactiver.
  #T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

  ctrl-O pour enregistrer et **ctrl-X** pour sortir.

- Réinitialiser inittab avec la commande init -q.
  pi@raspberrypi ~ $ sudo init -q

3 - Installation du module python minimalmodbus.

- Faire une mise à jour du RPi.
  pi@raspberrypi ~ $ sudo apt-get update

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

- Installer minimalmodbus.
  pi@raspberrypi ~ $ sudo pip install -U minimalmodbus

4 - Installation du module python intelhex.

- Télé-charger le code source.
  pi@raspberrypi ~ $ wget http://www.bialix.com/intelhex/intelhex-1.5.zip

- Décompresser le fichier.
  pi@raspberrypi ~ $ unzip intelhex-1.5.zip

- Installer le module.
  pi@raspberrypi ~ $ cd intelhex-1.5/
  pi@raspberrypi ~/intelhex-1.5 $ sudo python setup.py install

5 - Installation de git.

- Installer github.
  pi@raspberrypi ~/intelhex-1.5 $ cd
  pi@raspberrypi ~ $ sudo apt-get install git

6 - Installer mon code pour modbus.

- Télé-charger le code de github.
  pi@raspberrypi ~ $ git clone https://github.com/danjperron/NoContactWaterDetect

7- Installer BurnLVP. Le code python pour programmer la puce PIC.

- Télé-charger le code de github.
  git clone https://github.com/danjperron/burnLVP

8- Reboot

  sudo reboot

Étape 3 - Programmer La puce PIC

La puce PIC12F1840 à 4K de mémoire programme. Il est possible d'utiliser la RPi comme programmeur(brûleur) en mode LVP. Le code burnLVP.py va insérer le code binaire du fichier hex.

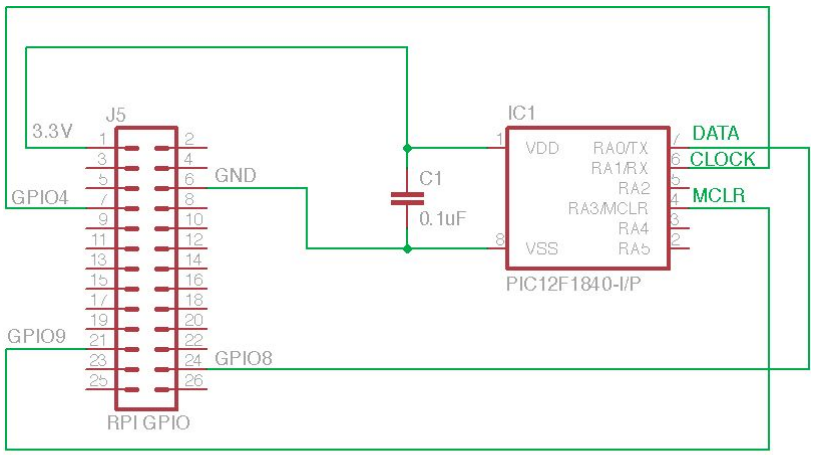


Figure 1. Schéma pour brûler la puce PIC avec le Raspberry Pi.

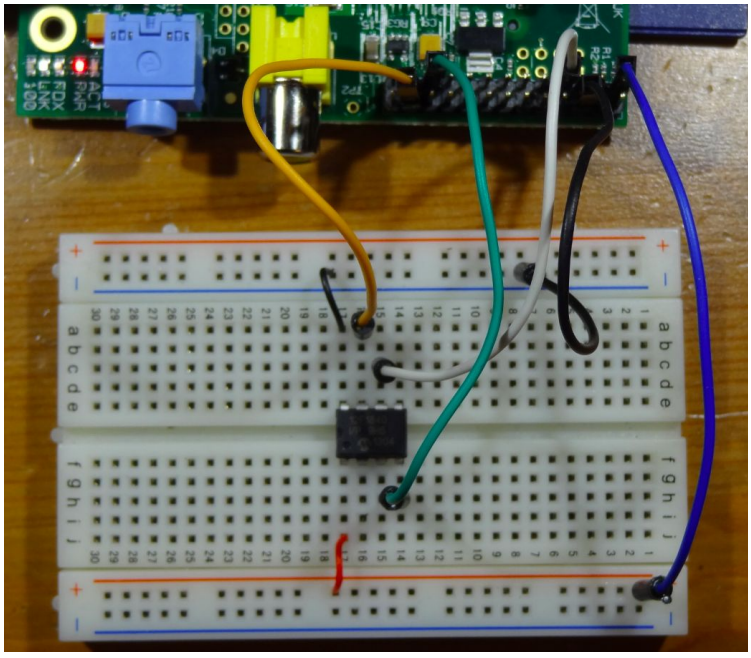


Photo 1. Programmation interne du PIC.

1 - Brûler le programme dans la puce PIC12F1840 pour le module de détection.

- Insérer la puce PIC12F1840 au montage du protoboard et brancher les fils selon le schéma de la figure 1.

- Exécuter l'application burnLVP.py avec le paramètre WaterDetectModbus.hex
  pi@raspberrypi ~ $ sudo ./burnLVP/burnLVP.py ./NoContactWaterDetect/WaterDetectModbus.hex

  La sortie console de l'application devrait être

```
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
```

- Débrancher le fil de tension 3.3V et enlever la puce.

- Marquer la puce avec une étiquette.

2 - Brûler le programme RS-485 de la même façon avec rs485Switch.hex.
pi@raspberrypi ~ $ sudo ./burnLVP/burnLVP.py ./NoContactWaterDetect/rs485switch.hex

Étape 5 - Vérification de la puce cpu

Après l'étape 4 , le cpu est maintenant fonctionnel. Il est maintenant possible de vérifier si cela a marcher. Pas besoin de RS-485, nous pouvons brancher la puce directement sur le RPi en utilisant les deux ports séries et le protocole Modbus.

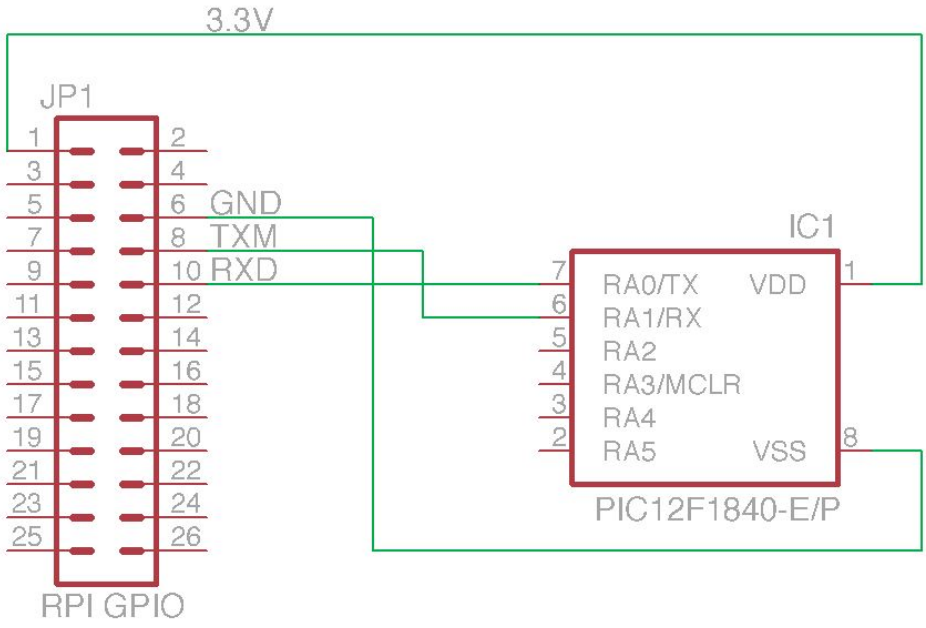
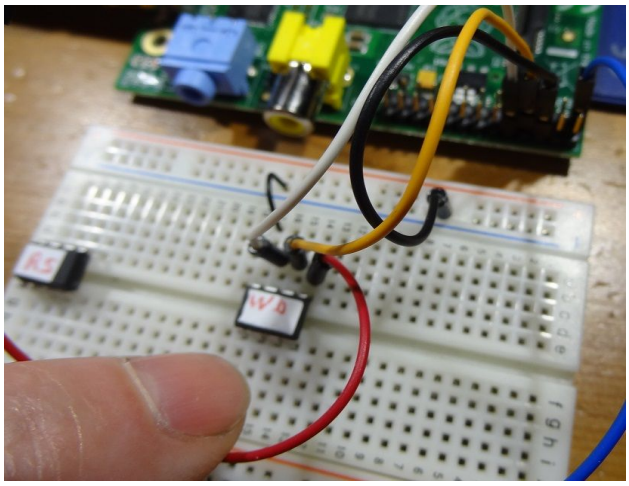


Figure 2. Mode de test pour vérifier la puce.

1 - Exécuter l'application "TestSlave127.py". la sortie console devrait être,

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 <- Je touche au fil ajouté sur RA2
F: 12243



Étape 5 - L’adresse esclave Modbus

Une fois que le code binaire est fraîchement enregistré dans le cpu, l’adresse Modbus par défaut est 127. Il est possible d’envoyer une commande modbus , fonction 6, adresse interne 2, pour modifier l’adresse Modbus.
J’ai créé une application python, SlaveAddress.py, à cette fin.

```
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 $
```

L’application python , SlaveAddress.py, nous informe que le module a retourné la mauvaise adresse. Et c’est exactement ce que nous voulions.
Peut être que je devrais corriger mais pour l’instant….

Étape 6 - Assemblage du prototype

Le montage complet sur le protoboard est la meilleur façon de tout vérifier.

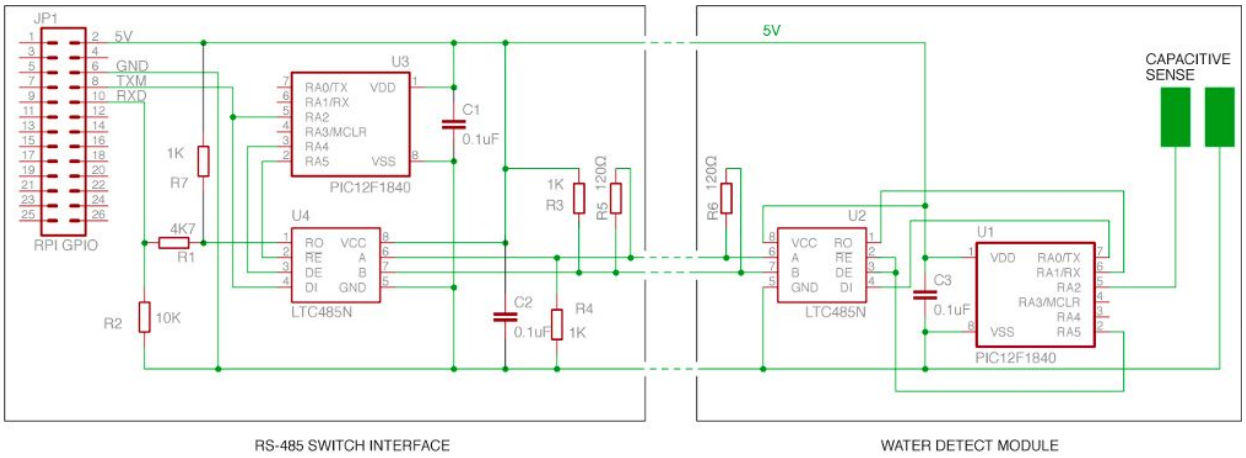


Figure 3 - Schéma de montage sur protoboard.

Un fois tout monter , l'application python CheckModbus.py devrait fonctionner. N'oublier pas de changer l'address Modbus! La prochaine étape sera de changer l'alimentation et d'ajouter le régulateur.

P.S. Ajouter un fil sur la broche 5 du cpu et toucher le pour simuler l'eau.

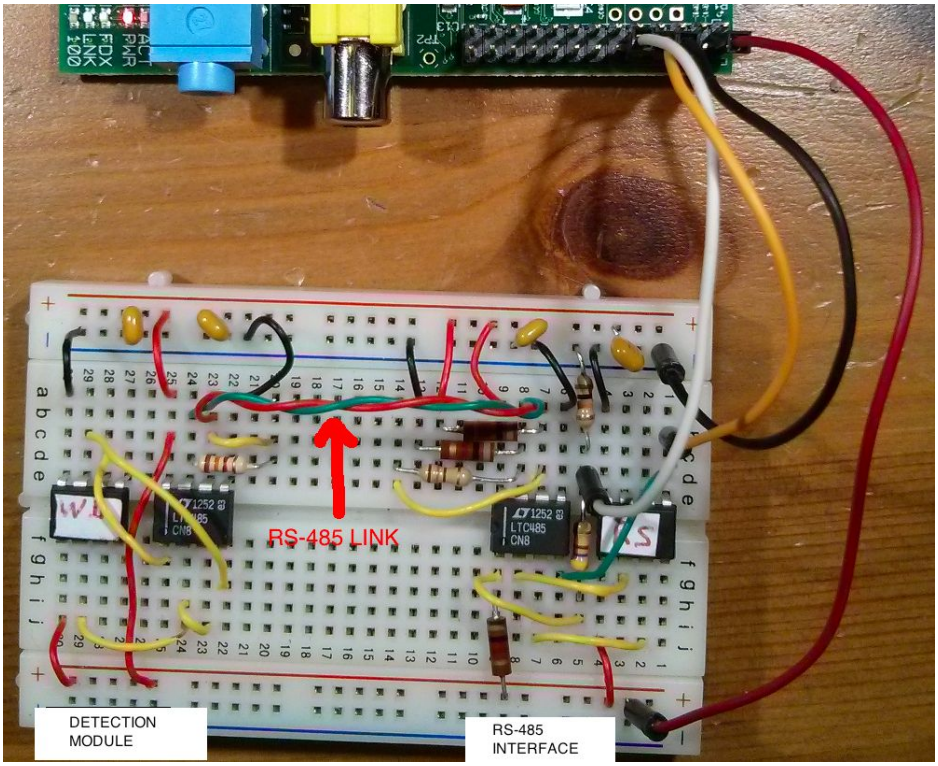


Photo 3 - Le montage sur protoboard.

Étape 7 - Assemblage du détecteur

La première chose à faire est de couper le veroboard pour les dimensions du boîtier. Évidement il faut que tout le montage soit capable d'être installé. J'utilise un "socket" pour les puces. De cette façon je peux facilement reprogrammer le cpu. Le schéma est dans l'annexe, figure 5.

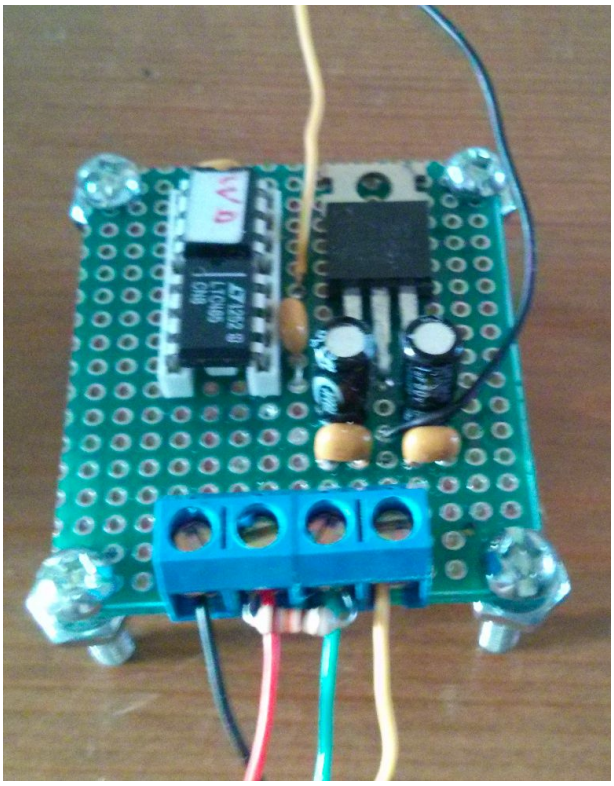


Photo 4 - Le circuit du détecteur.

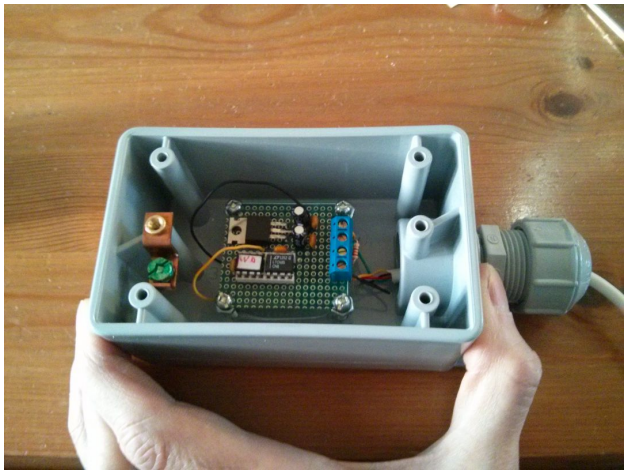


Photo 5- Le détecteur dans son boîtier.

Le détecteur lui-même consiste de 2 rubans d'aluminium autocollants de chaque côté du tube. Dénudez les fils et les déposez en zigzag. Avec un autre ruban d'alluminium consolidez le tout.

Photo 6- Le fil en zigzag pour un meilleur contact.

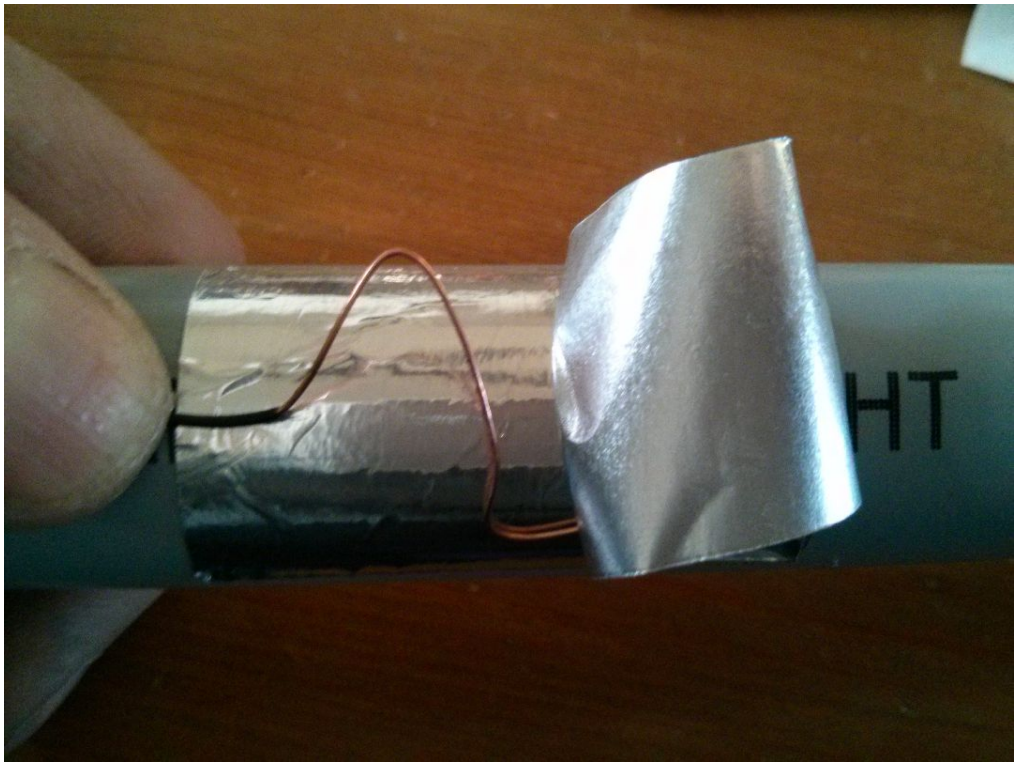


Photo 7- Le fil en sandwich pour solidifier le tout.

Photo 8- Il ne manque qu’un adaptateur pour relier le tout.

Un tube auto-rétrécissant, (heat shrink tube), avec du silicon à chaque bout devrait sceller le tout.

Étape 8 - Interface page Web

J'utilise l'application python webiopi qui est un serveur web. Il est facile d'y ajouter des fonctions. J'ai donc ajouté les fonctions pour lire et écrire les registres Modbus.

Pour installer webiopi, suivez les instruction sur la page
https://code.google.com/p/webiopi/wiki/INSTALL

ou faites,

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

Veuillez modifier le javascript des fichiers index.fr.html et ModbusSettings.fr.html pour votre environnement spécifique.

Les fichiers sont dans le github (~/NoContactWaterDetect) .

Pour démarrer l'application serveur web , exécutez

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

Et maintenant ouvrez votre navigateur sur l'url

http://<Rasberry PI addresse IP>:8000/index.fr.html

P.S. Pour trouver l'IP utilisez, `sudo ifconfig` et cherchez 'inet addr:' sur eth0 ou wlan0 si vous avez un wifi.

Le fichier index.fr.html utilise javascript pour automatiser la création de la table des modules. Veuillez modifier la variable ‘ValidModuleList’ pour que la liste corresponde avec votre configuration.

Voici un exemple de la page Web

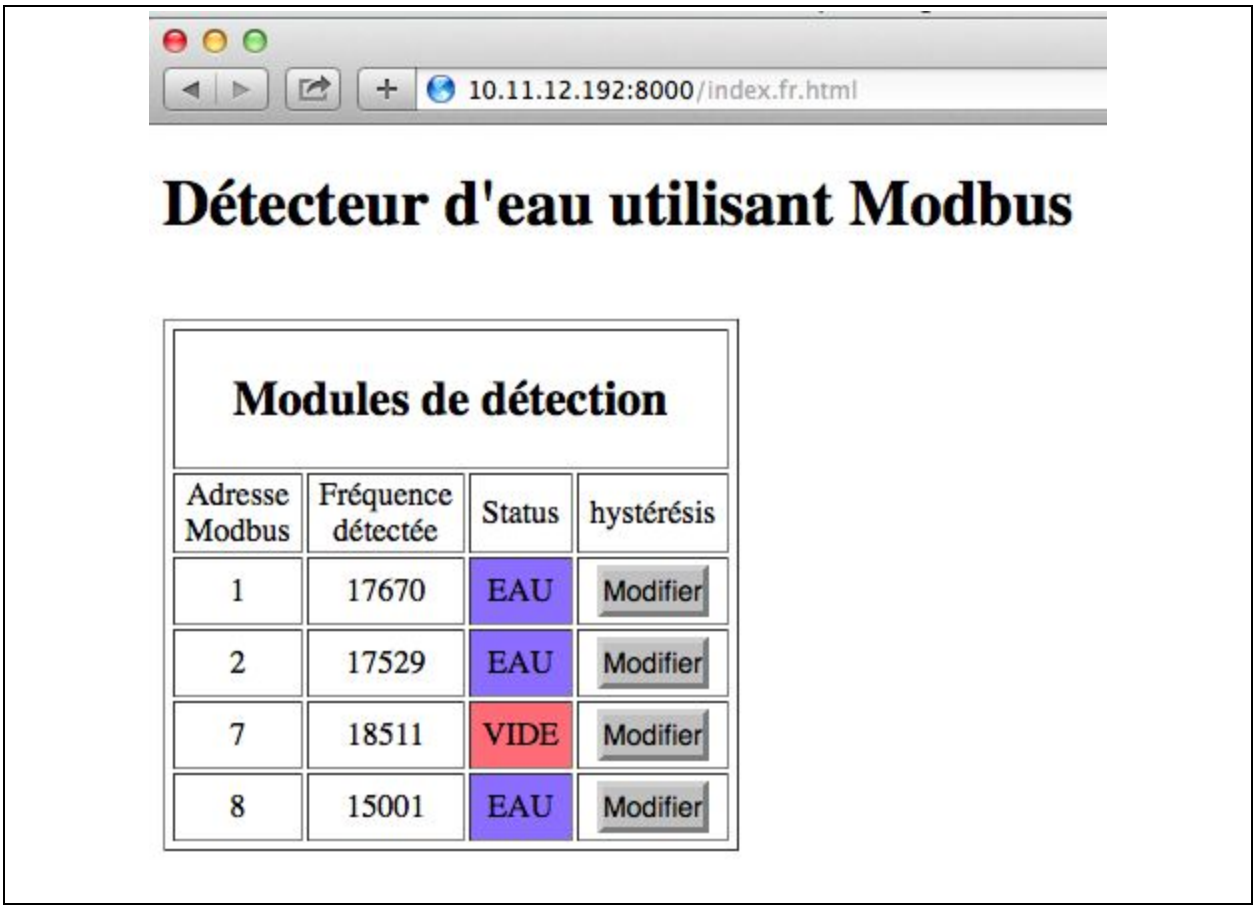


Figure 4 - Interface web avec quatre modules

Conclusion

J’espère que ce petit quide vous donnera le gout d’expérimenter avec l’électronique.

Je vous laisse le soin de compléter l’interface RS-485.

Maintenant que vous êtes familier avec Modbus, vous pouvez ajouter des modules entrées/sortie, des Convertisseur analogique, des compteurs de fréquence, etc…

Bon Montage,

Daniel Perron

Référence

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

ANNEXE 1

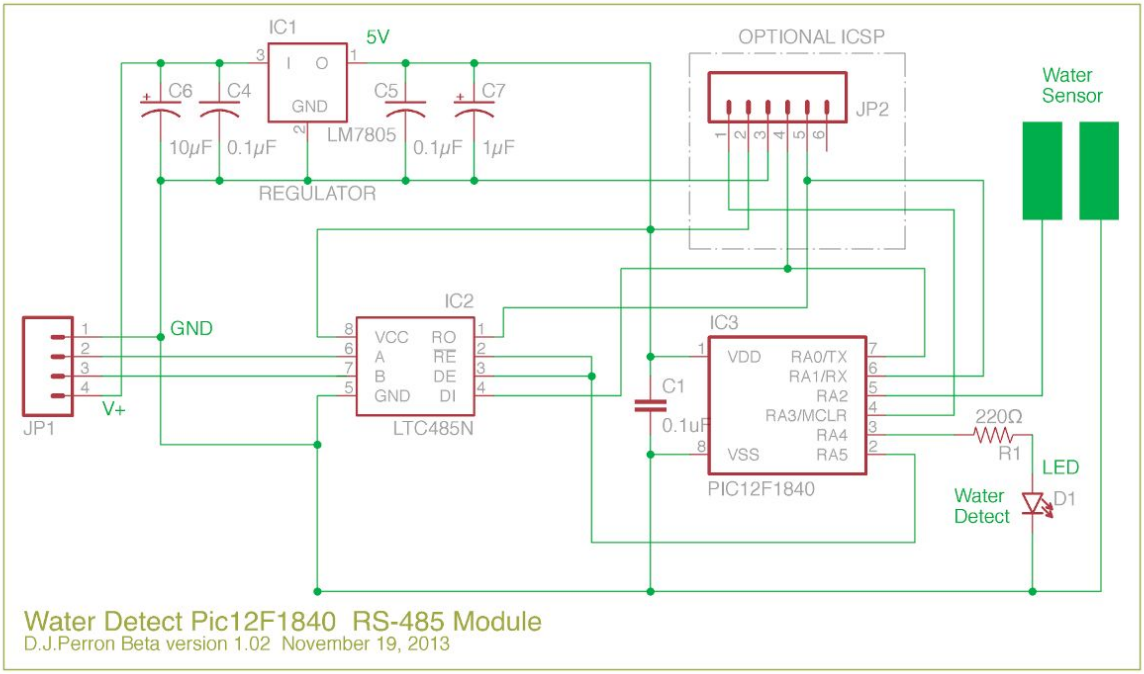


Figure 5 - Module de détection de liquide.

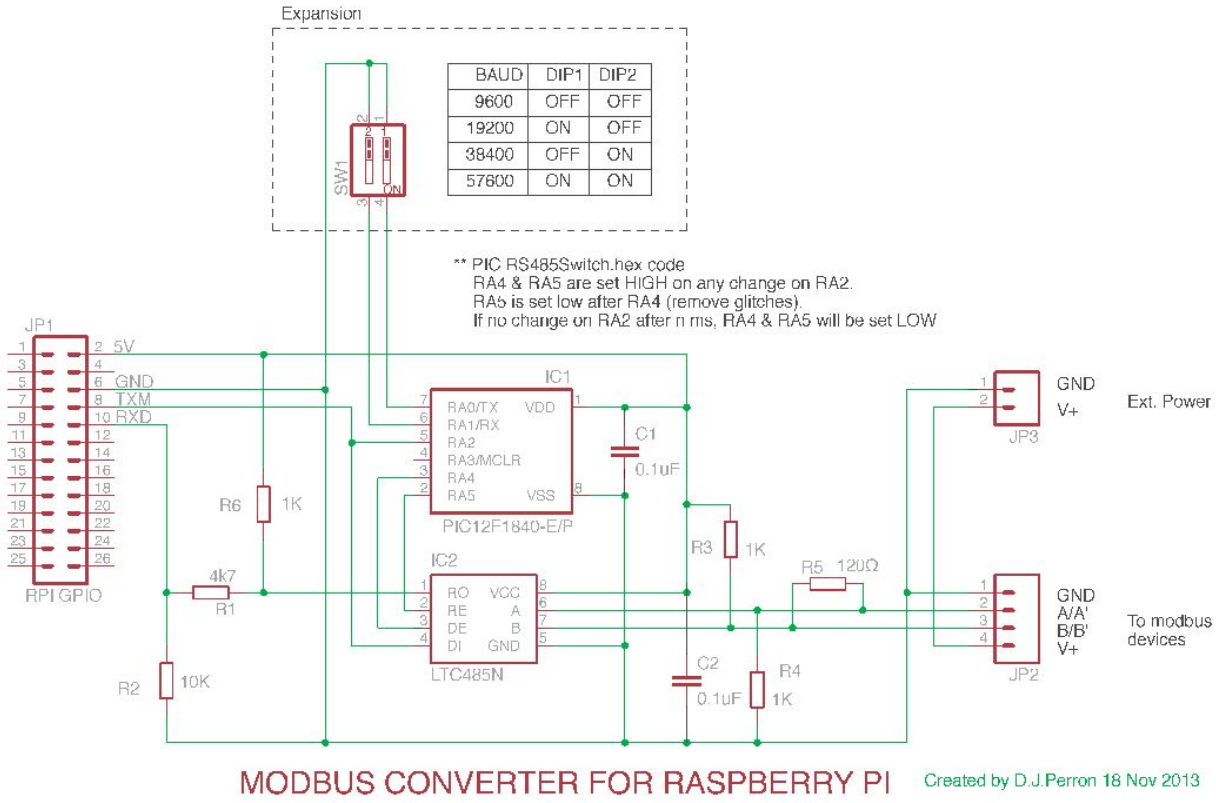
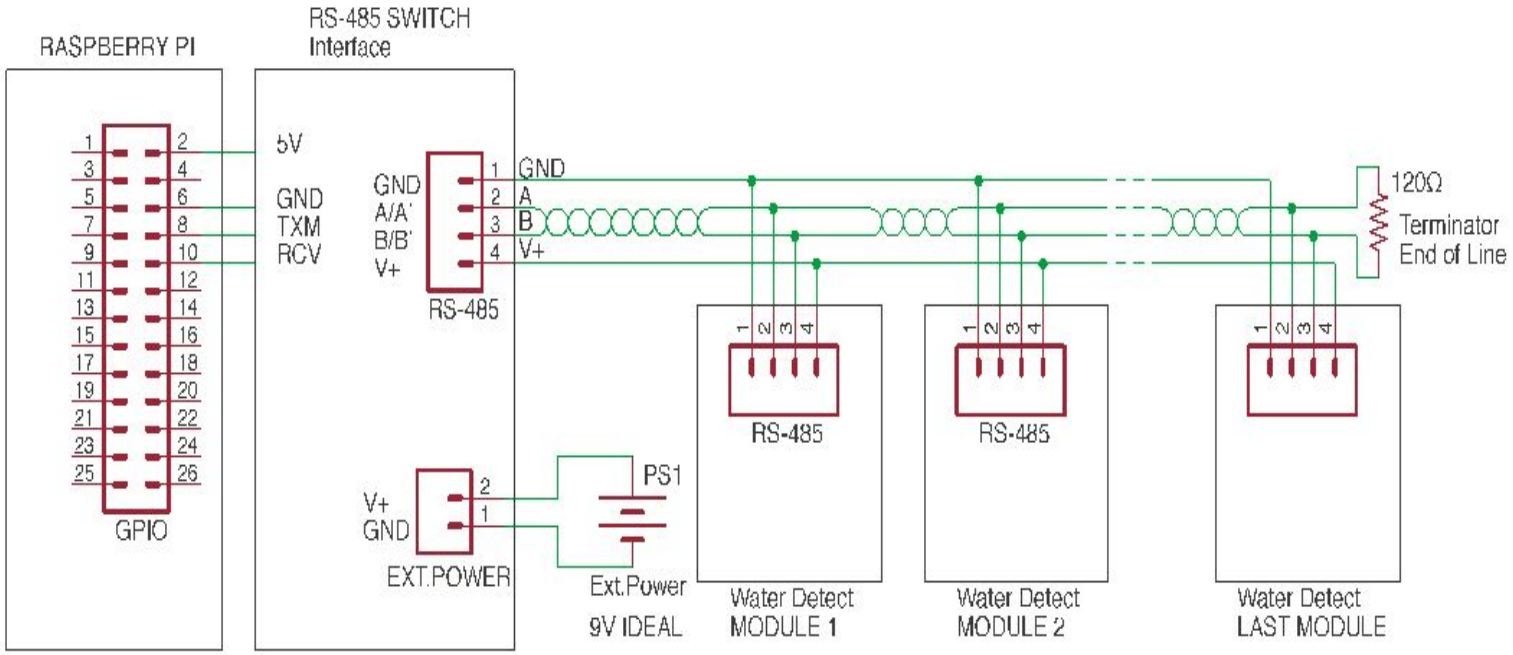


Figure 6 - Interface RS-485 commuté.



Raspberry PI Modbus layout with Water Detect modules.

D.J.Perron 19 November 2013

Figure 7 - Schéma bloc du système complet