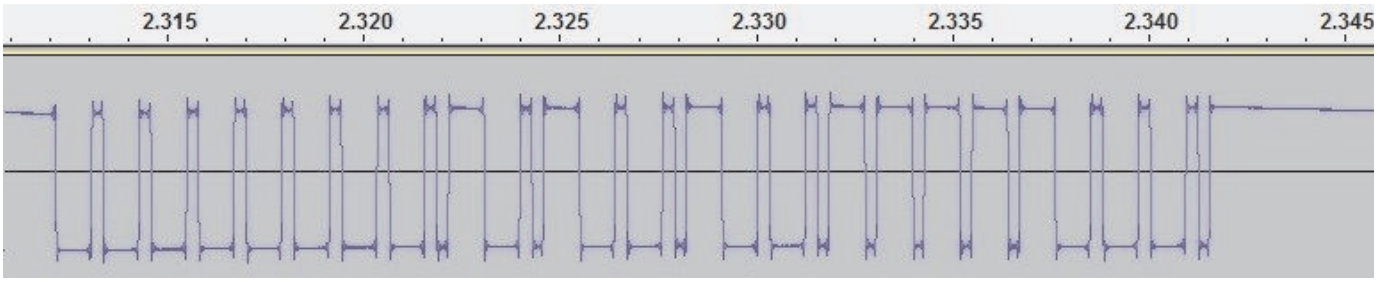
Pour lire le code émis par la télécommande, j'ai branché le module récepteur sur la prise microphone d'un ordinateur. J'ai utilisé une alimentation 5V séparée, mais il y a une sortie 5V sur le connecteur GPIO du Raspberry Pi qui peut servir à cet usage. La sortie du module récepteur est du 5V numérique, ce qui ne convient pas à une connexion directe sur la prise audio de l'ordinateur. J'ai donc branché la sortie du module à la prise microphone de mon portable via une résistance de 1M Ohm. J'ai assemblé le circuit sur une platine, mais vous pouvez utiliser l'autre extrémité du câble de lecteur de disquettes cité dans la section Matériel.

Audacity (http://audacity.sourceforge.com) est un excellent gratuiciel pour examiner des signaux de ce genre. Une fois satisfait par le niveau d'enregistrement que je trouvais relativement correct, j'ai démarré l'enregistrement et appuyé sur l'un des boutons de la télécommande. Ayant arrêté l'enregistrement, j'ai été en mesure d'agrandir



la région dans laquelle se trouvait le signal de la commande. Le signal se répétait encore et encore jusqu'à ce que le bouton soit relâché. Les impulsions étroites semblaient durer environ 0,25 ms tandis que les larges duraient 3 fois plus longtemps.

J'ai créé cette chaîne binaire à partir de la forme d'onde. Chaque bit représente 0,25 ms avec 1 = impulsion haute et 0 = impulsion basse. Pour faciliter la lecture, les impulsions ont été séparées par des tirets.

```
11111-000-1-000-1-000-1-000-1-000-1-000-
1-000-1-000-1-0-111-000-1-0-111-000-1-
000-1-0-111-000-1-000-1-0-111-0-111-0-
111-0-111-0-111-000-1-000-1-000-1-0-
111111
```

Pour le projet, l'heure est venue d'agir ou de s'arrêter. Si vous êtes arrivés à ce point et qu'il vous est impossible de trouver un motif qui se répète en apparaissant lorsque vous appuyez sur les boutons de la télécommande, cela signifie que vos prises peuvent ne pas utiliser de simples signaux AM sur lesquels repose ce projet.

Envoi d'un signal

Pour pouvoir exploiter les données capturées, j'ai branché l'émetteur sur le Raspberry Pi. Comme il est prévu de le laisser connecté, il est par conséquent alimenté par la broche +5V du GPIO. J'ai relié GPIO 7 depuis le Raspberry Pi à la broche de données et la masse de l'émetteur à celle du connecteur GPIO du Raspberry. Les signaux 3,3V sont suffisants pour piloter l'émetteur, quoique je n'ai découvert ça qu'en faisant le test.

Linux étant un système d'exploitation multi-tâches, quelque chose peut très bien utiliser le CPU au mauvais moment, c'est pourquoi mon programme envoie la séquence de bits à 10 reprises dans l'espoir qu'au moins une soit transmise correctement.

Quand j'ai lancé mon logiciel sur mon Raspberry Pi pour la première fois, j'ai capturé le signal avec Audacity. J'ai pu voir que la surface de la forme d'onde était correcte mais à l'envers. Inutile de dire que la prise ne réagissait pas. J'ai donc inversé tous les bits de mon flux de sortie et relancé le code de test. Cette fois la prise s'est allumée ! Il ne me restait plus qu'à décoder les autres boutons.

Matériel

Le connecteur pour le GPIO et le câble que j'ai utilisés ont été faits à partir de câbles pour lecteur de disquettes de vieux PC. Ils sont plus larges que le connecteur GPIO du Raspberry Pi, mais vous pouvez placer la prise sur les broches du GPIO en vous calant sur un bord. Bien sûr, ça ne sera pas possible si votre Raspberry Pi est dans un boîtier.

Trois fils seulement sont nécessaires pour le GPIO : +5V, GND et GPIO 7 (CE1) pour la broche que j'utilise pour contrôler l'émetteur.

Un tout petit morceau de plaque d'essai est placé à l'autre extrémité de la nappe. Soudé