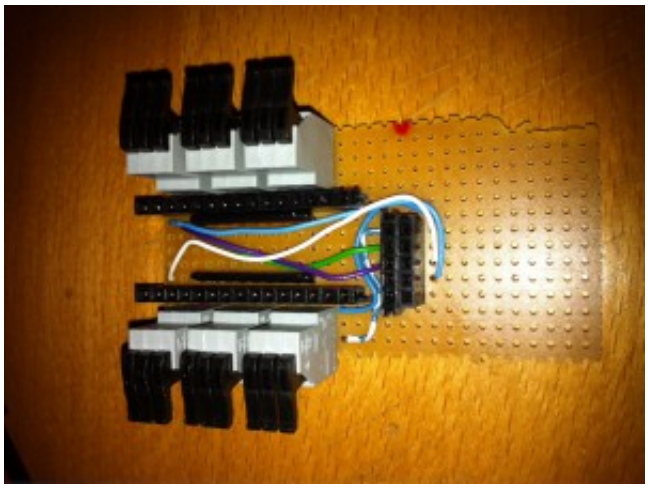
# Arduino & impulsions (ou fenêtres)

Continuons la série avec de l'arduino.

Voilà l'exemple typique du montage basique pouvant servir à plein de choses. Moi, il me sert d'une part à compter les impulsions de mes sous-compteurs électriques (Voltcraft Geco32) pour connaître la conso précise de chaque ligne de l'installation et sera, plus tard, également dévolu à la surveillance des fenêtres et plus généralement de tout objet communiquant en fermant ou en ouvrant un contact (détecteur de mouvement, d'inondation, sonnette…)



Le principe est tout bête. Il faut surveiller les broches de l'arduino très rapidement en boucle pour détecter les changements (impulsions dans le premier cas ou évènement plus long si on détecte une ouverture de fenêtre). Mes tests ont montré que l'arduino parvient sans encombre à compter des impulsions de 250ms.

Matériellement comment ça se passe ? On va utiliser toutes les broches imaginables (sauf 10, 11, 12 et 13 qui vont servir plus tard) ce qui, en gardant 0 et 1 de côté, nous fait quand même 16 lignes et on va les brancher, à travers une résistance pour chacune (j'ai pris du 10kohm en deux réseaux de 8), vers la GND de l'arduino pour s'assurer que lorsqu'il n'y a pas de courant elle sont bien à 0. De l'autre côté, on va les relier au +5v par l'intermédiaire du montage détectant les évènements, montage qu'on peut bêtement considérer comme étant un interrupteur.

Du coup, quand l'interrupteur est en position fermée, le +5v passe jusqu'à la broche avant d'aller mourir dans la GND à travers la résistance (si on ne la met pas, PAF l'arduino) et la broche intercalée au milieu détecte un état haut. Si on ouvre l'interrupteur, la broche descend au niveau bas.

Sur la photo ci-dessus, vous voyez les borniers de chaque côté avec 8 entrées et la sortie +5v commune tout à droite sur le dernier bornier. L'arduino (nano v3) vient s'enficher sur les deux connecteurs horizontaux, et entre les deux, vous voyez les deux réseaux de résistances.

Le connecteur sur la droite vient recevoir un module ethernet (utilisant les fameuses broches 10

à 13 de tout à l'heure) pour rendre le montage communiquant sur un réseau IP. J'en parlerais dans un futur article.

Un peu de programmation avancée maintenant. Pour détecter des ouvertures, une simple boucle for vous suffira, vous devriez vous en sortir avec le site arduino.cc pour avancer. Pour compter de courtes impulsions, c'est peut-être un peu plus coton. Voici le code (sale) que j'ai pondu rapidement :

```
unsigned long count[16];
byte recvN;
byte recvA;
byte prevN;
byte prevA;
byte buff;
byte pbuff;

void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  byte charac;
  byte prevcharac;
  int shoot = 0;

  recvN = PIND >;>; 2;
  charac = PINB ; 1;
      buff = buff >;>; 1;
    }
    prevN = recvN;
  }

  recvA = PINC;
  if (recvA != prevA) {
    buff = recvA;
    pbuff = prevA;
    for (int i = 8 ; i       charac = buff & 1;
      prevcharac = pbuff & 1;
      if ( (charac == 0) && (prevcharac == 1) ) {
         count[i]++;
         shoot=1;
      }
      pbuff = pbuff >;>; 1;
      buff = buff >;>; 1;
    }
   prevA = recvA;
```

```
  }
  if (shoot == 1) {
    Serial.print("(");
    for (int i = 0 ; i       Serial.print(count[i]);
      Serial.print(";");
    }
    Serial.println(")");
  }
}
```

Quelques explications. Il y a trois grandes parties dans le code : la gestion de la volée de broches numériques, la volée analogique et la notification sur le port série de l’arduino. Pour aller plus vite et ne pas lire une par une les broches, je me suis servi des variables PINB, PINC et PIND qui contiennent chacune un octet rassemblant l’état de 8 broches, une sur chaque bit. Quelques rotations de bit et additions sont nécessaires sur PIND et PINB pour éliminer les broches 0 et 1 et les broches 10 à 13. On se retrouve finalement avec les broches de 2 à 9 ce qui tombe bien puisque dans un octet nous avons 8 bits.

On fait la même chose avec la variable PINC qui contient l’état des broches analogiques. Le fonctionnement est simple, on observe l’état de chaque bit et s’il était avant à 1 et qu’il vient de passer à 0 (front descendant) on incrémente le compteur et on positionne la variable shoot à 1 ce qui permet de n’en mettre sur le port série de l’arduino que lorsqu’il y a effectivement une impulsion.

Au prochain épisode j’essaie de vous toucher deux mots du module ENC28J60 qui permet de faire causer l’arduino sur un réseau IP pour deux francs six sous.