=7

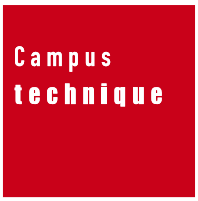
Laboratoire d'Arduino : Rapport

1er Bachelier en Informatique

Electricité et élément d'électronique pour l'interfaçage informatique

Auteur :
Timothée Simon

Enseignant :
David Arnaud



Année académique 2016 - 2017

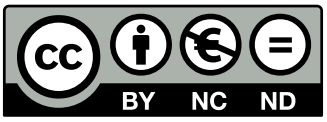


Table des matières

Chapitre 1

Introduction

Dans ce rapport nous allons utiliser un Arduino Uno ainsi que toute une série de modules compris dans un coffret. Arduino est une plateforme de développement open source via des cartes de développement. Puisque le projet est open source, les schemas des cartes est disponnible gratuitement ainsi que la suite logicielle Arduino que nous avons utilisé dans ces manipulations. L'Arduino que nous utilisons est l'Arduino Uno qui possède 14 entrés/sortie numérique dont 6 sont compatibles PWM et 6 entrées/sorties analogiques. Il possède également une prise USB pour injecter notre code ou pour l'alimenter et une prise d'alimentation jack. Efin, un bouton reset est présent pour 'rebooter' l'Arduino. Pour la programation de l'Arduino nous

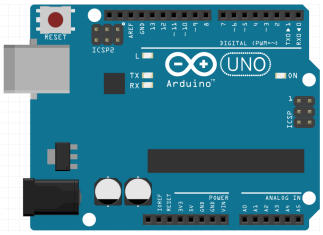


FIGURE 1.1 – Arduino Board

utillisons la suite officielle d'Arduino disponible gratuitement. Pour les schematiques nous avons utilisé Fritzing, également fournis gratuitement par la fondation Arduino.

FIGURE 1.2 – Suite Arduino

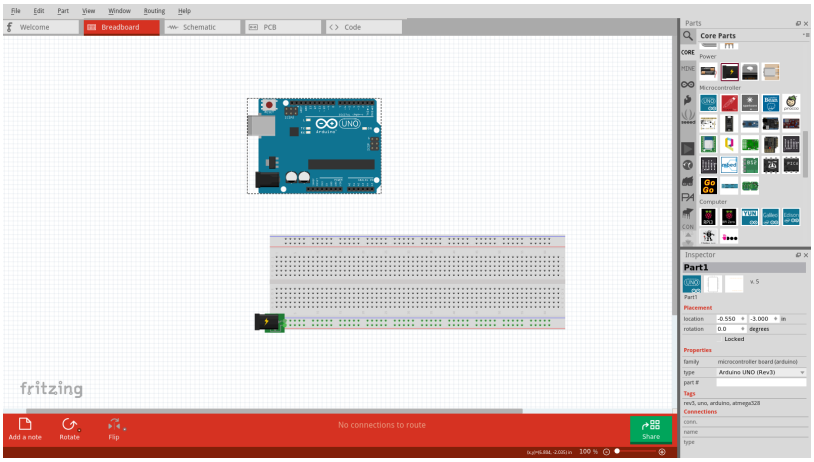


FIGURE 1.3 – Fritzing

Chapitre 2

Manipulations

1 TP1 : Bling Bling

1.1 Objectif

Dans ce TP nous allons commencer par faire clignoter une LED, puis nous utilserons la led RGB poru afficher une couleur précise, finalement nous ferons un compte à rebour avec l'afficheur 7 segments et l'afficheur 4 fois 7 segments.

1.2 Matériel

— Un ordinateur

— Un arduino Uno R3

— Une LED simple

— Une LED RGB

— Un afficheur 7 segments

— Un afficheur 4 fois 7 segments

— Une résistance de 220Ω

1.3 LED clignotante

Dans cette manipulation nous allons faire clignotter une LED en la branchant au port 13 de l'arduino. Ce port correspond également à la LED inclue dans l'arduino, nous pourrions donc même ici nous passer de la breadboard. Nous mettons bien

sur aussi une résistance de 220Ω en série avec la LED, mais nous aurions aussi pu utiliser la résistance *pullup*.

```
// Utilisation du port 13 pour la LED rouge
const int led = 13;

void setup() {
  // Initialise le port en tant que sortie
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);    // Allume la LED
  delay(1000);                // Attend 1000 ms
  digitalWrite(led, LOW);     // Eteint la LED
  delay(1000);                // Attend 1000 ms
}
```

Code/TP1/TP1.1/TP1.1.ino

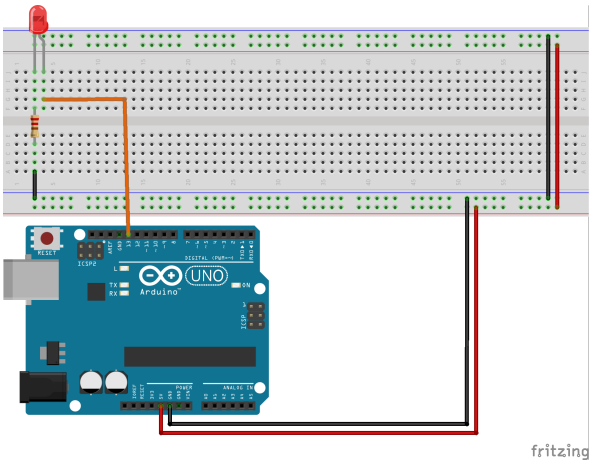


FIGURE 2.1 – LED clignotante

1.4 LED RGB

Dans cette manipulaton nous allons controller la couleur d'une LED RGB qui est en faite composée de 3 LED, une rouge, une verte et une bleu. Nous relions donc les pattes correspondantes aux couleurs à notre arduino et la patte correspondant à la cathode commune au +VCC via une résistance. Le code correspondant à une LED RGB à annode commune est aussi inclu. Nous utilisons les ports analogiques et non digitaux pour pouvoir controler l'intensité de chaque couleur et ainsi allumer la LED de la couleur souhaitée.

```
const int redPin = 11; // Utilisation du pin 11 pour la couleur rouge
const int greenPin = 10; // Utilisation du pin 10 pour la couleur verte
const int bluePin = 9; // Utilisation du pin 9 pour la couleur bleu

void setup()
```

```
{
  // Initialisation des pins comme sortie
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop()
{
  setColor(255, 0, 0);  // Affiche du rouge pendant une seconde
  delay(1000);
  setColor(0, 255, 0);  // Affiche du vert pendant une seconde
  delay(1000);
  setColor(0, 0, 255);  // Affiche du bleu pendant une seconde
  delay(1000);
  setColor(255, 255, 0);  // Affiche du jaune pendant une seconde
  delay(1000);
  setColor(80, 0, 80);  // Affiche du mauve pendant une seconde
  delay(1000);
}

void setColor(int red, int green, int blue)
{
  // A decommenter si la led est a anode commune
  /*
     red = 255 - red;
     green = 255 - green;
     blue = 255 - blue;
   */
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

Code/TP1/TP1.2/TP1.2.ino

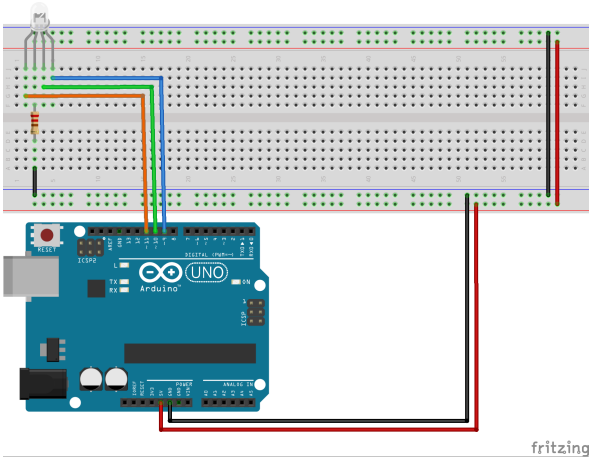


FIGURE 2.2 – LED RGB

1.5 Afficheur 7 segments simple

Dans cette manipulation nous allons connecter un afficheur 7 segments à un notre arduino pour lui faire afficher un compte à rebour allant de 9 à 0. Alternativement nous aurions pu lui faire aire un compte à rebour alant de F (16 en hexadécimal) à 0.

```
const int a = 2;  // Le segment a est connecte au pin 2
const int b = 3;  // Le segment a est connecte au pin 3
const int c = 4;  // Le segment a est connecte au pin 4
const int d = 5;  // Le segment a est connecte au pin 5
const int e = 6;  // Le segment a est connecte au pin 6
const int f = 7;  // Le segment a est connecte au pin 7
const int g = 8;  // Le segment a est connecte au pin 8
const int dot = 9; // Le segment correspondant au point est connecte au
    pin 9

void setup() {
  // Initialisation de tout les segment en tant que sortie
  pinMode(a, OUTPUT);  //A
  pinMode(b, OUTPUT);  //B
  pinMode(c, OUTPUT);  //C
  pinMode(d, OUTPUT);  //D
  pinMode(e, OUTPUT);  //E
  pinMode(f, OUTPUT);  //F
  pinMode(g, OUTPUT);  //G
}

void loop() {
  for(int number = 9; number >=0; number--) // Commence un décompte
   allant de 9 à 0 toute les seconde
  {
    ClearNumber();
    displayNumber(number);
    delay(1000);
  }
  //delay(1000);
}

void displayNumber(int number)
{
  // Affiche le chiffre souhaité sur l'afficheur 7 segments
  if(number != 1 && number != 4 && number != 11 && number != 12 &&
   number != 13)
    digitalWrite(a,HIGH);

  if(number != 5 && number != 6 && number != 11 && number != 12 &&
   number != 14 && number != 15)
    digitalWrite(b,HIGH);

  if(number !=2 && number != 12 && number != 14 && number != 15)
    digitalWrite(c,HIGH);

  if(number != 1 && number != 4 && number != 7 && number != 10 &&
   number != 15)
```

```
    digitalWrite(d,HIGH);

  if(number == 2 || number == 6 || number == 8 || number == 0 || number
    >= 10)
    digitalWrite(e,HIGH);

  if(number != 1 && number != 2 && number != 3 && number != 7 && number
    != 12 && number != 13)
    digitalWrite(f,HIGH);

  if (number != 0 && number != 1 && number !=7)
    digitalWrite(g,HIGH);


}

void ClearNumber()
{
  // Réinitialise l'afficheur 7 segments
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
  digitalWrite(g,LOW);
}
```

Code/TP1/TP1.3/TP1.3.ino

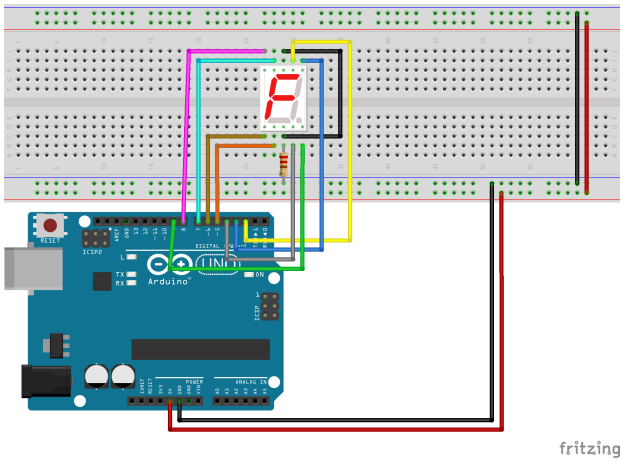


FIGURE 2.3 – Afficheur 7 segments

1.6 Afficheur 7 segments quadruple

```
const int a = 2;  // Le segment a est connecte au pin 2
const int b = 3;  // Le segment a est connecte au pin 3
const int c = 4;  // Le segment a est connecte au pin 4
const int d = 5;  // Le segment a est connecte au pin 5
const int e = 6;  // Le segment a est connecte au pin 6
```

```
const int f = 7;  // Le segment a est connecte au pin 7
const int g = 8;  // Le segment a est connecte au pin 8
const int dot = 9; // Le segment correspondant au point est connecte au
    pin 9
const int d1 = 10; // Le premier chiffre est connetce au pin 10
const int d2 = 11; // Le second chiffre est connetce au pin 11
const int d3 = 12; // Le troisieme chiffre est connetce au pin 12
const int d4 = 13; // Le dernier chiffre est connetce au pin 13

void setup() {
  // Initialisation de tout les segment en tant que sortie
  pinMode(a, OUTPUT);  //A
  pinMode(b, OUTPUT);  //B
  pinMode(c, OUTPUT);  //C
  pinMode(d, OUTPUT);  //D
  pinMode(e, OUTPUT);  //E
  pinMode(f, OUTPUT);  //F
  pinMode(g, OUTPUT);  //G
  pinMode(d1, OUTPUT);  //d1
  pinMode(d2, OUTPUT);  //d2
  pinMode(d3, OUTPUT);  //d3
  pinMode(d4, OUTPUT);  //d4
}

void loop() {
  for(int number = 9999; number >=0; number--) // Commencer le décompte
    à 9999, le finir à 0
  {
    for(int i = 0; i <1001; i++) // Rafraichir l'écran toute les
    millisecondes pendant 1 seconde
    {
      displayNumber(number); // Afficher le nombre
      delay(1);
    }
  }
  delay(1000);
}

void displayNumber(int number)
{
  ClearNumber();
  displayDigit(number%10, 0); // L'unité correspond au nombre%10

  ClearNumber();
  displayDigit((number%100)/10, 1); // La disaine correspond au nombre
    %100 divisé par 10 par division entière

  ClearNumber();
  displayDigit((number%1000)/100, 2); // La centaine correspond au
    nombre%1000 divisé par 100 par division entière

  ClearNumber();
  displayDigit(number/1000, 3); // Le millier correspond au nombre
    divisé par 1000 par division entière
}
```

```
56 void displayDigit(int number, int pos)
{
58   // Active la position souhaitée
   if(pos == 0)
60     digitalWrite(d1, HIGH);

62   if(pos == 1)
     digitalWrite(d2, HIGH);
64
   if(pos == 2)
66     digitalWrite(d3, HIGH);

68   if(pos == 3)
     digitalWrite(d4, HIGH);
70
   // Allume les segments correspondants au chiffre souhaité
72
   if(number != 1 && number != 4 && number != 11 && number != 12 &&
    number != 13)
74     digitalWrite(a,HIGH);

76   if(number != 5 && number != 6 && number != 11 && number != 12 &&
    number != 14 && number != 15)
     digitalWrite(b,HIGH);
78
   if(number !=2 && number != 12 && number != 14 && number != 15)
80     digitalWrite(c,HIGH);

82   if(number != 1 && number != 4 && number != 7 && number != 10 &&
    number != 15)
     digitalWrite(d,HIGH);
84
   if(number == 2 || number == 6 || number == 8 || number == 0 || number
     >= 10)
86     digitalWrite(e,HIGH);

88   if(number != 1 && number != 2 && number != 3 && number != 7 && number
     != 12 && number != 13)
     digitalWrite(f,HIGH);
90
   if (number != 0 && number != 1 && number !=7)
92     digitalWrite(g,HIGH);


94
}
96
void ClearNumber()
98 {
   // Passe tout les sorties à 0 et reset l'afficheur.
100  digitalWrite(d1, LOW);
   digitalWrite(d2, LOW);
102  digitalWrite(d3, LOW);
   digitalWrite(d4, LOW);
104  digitalWrite(a,LOW);
   digitalWrite(b,LOW);
106  digitalWrite(c,LOW);
```

```
    digitalWrite(d,LOW);
108 digitalWrite(e,LOW);
    digitalWrite(f,LOW);
110 digitalWrite(g,LOW);
}
```

Code/TP1/TP1.4/TP1.4.ino

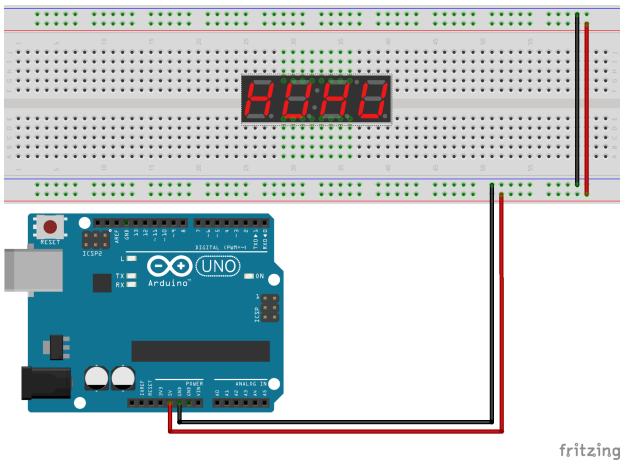


FIGURE 2.4 – Afficheur 7 segments quadruple

2 TP2 : LCD Time

2.1 Objectif

Dans cette manipulation nous allons apprendre a utiliser un ecran LCD. Nous commencerons par decouvrir comment ecrire des phrases simples, puis nous afficherons un texte defillant et enfin nous utiliserons des variables pour afficher l'heure.

2.2 Matériel

- Un ordinateur
- Un arduino Uno R3
- Un ecran LCD
- Une resistance de 10KΩ
- Un potentiometre

2.3 Nom-Prenom

Le montage est identique pour toutes les manipulations de ce labo. Nous avons utilise la librairie LiquidCrystal, disponnible sur le site officiel d'Arduino. Ici nous allons écrire nos noms sur chacune des lignes de l'écran LCD avec, à gauche, le numéro de notre groupe. Nous utilsons la librairie LiquidCrystal qui nous permet de facilement afficher du texte sur notre écran LCD.

```
// Inclu la librairie LiquidCrystal qui permet de controller l'écran
#include <LiquidCrystal.h>

// Réglages de LiquidCristal par rapport aux pins utilisés
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // Initialise un écran de 2 lignes de 16 caractères
  lcd.begin(16, 2);
}


void loop() {
  // Affiche mon nom et le chiffre 4 conne dernier caractère
  lcd.print("Timothee Simon 4");
  // Commence à écrire sur la deuxième ligne
  lcd.setCursor(0, 1);
  // Affiche mon nom et le chiffre 4 conne dernier caractère
  lcd.print("John Snow      4");
  delay(1);
}
```

Code/TP2/TP2.1/TP2.1.ino

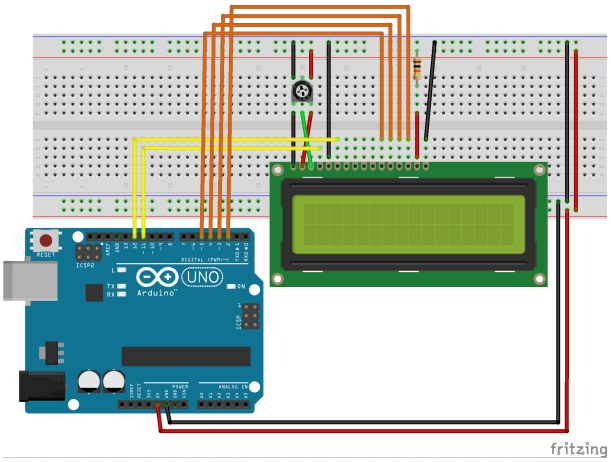


FIGURE 2.5 – Nom-Prenom

2.4 Texte deroulant

Dans cette manipulation nous allons afficher un texte déroulant vers la gauche.

```
// Inclu la librairie LiquidCrystal pour controller l'écran
#include <LiquidCrystal.h>

// Initialise les ins correspondants à l'écran
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // Initialise un écran de deux lignes de 16 charactères
  lcd.begin(16, 2);
  // Afficher mon nom
  lcd.print("Timothee Simon");
}

void loop() {
  // Décale le contenu de l'écran d'un charatère à gauche
    lcd.scrollDisplayLeft();
  // Attend 150 ms
    delay(150);
}
```

Code/TP2/TP2.2/TP2.2.ino

2.5 Horloge

Dans cette manipulation nous allons afficher une horloge au milieu de la deuxième ligne de l'écran. Une fois téléversé, l'Arduino commencera a afficher l'heure à 22h59m50

car il n'est pas possible d'afficher l'heure réelle sans utiliser des boutons pour régler l'heure ou un module *Real Time Clock*.

```
// TODO Ordonner initialisaion de l'heure et
// Inclu la librairie LiquidCrystal qui permet de controler l'écran
#include <LiquidCrystal.h>

//  Réglages de LiquidCristal par rapport aux pins utilisés
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // Initialise un écran de 2 lignes de 16 caractères
  lcd.begin(16, 2);
}

void loop() {
  // Initialise l'heure à 22h59m50 et le décalage à 4 characères
  int heure = 22, minutes = 59, secondes = 50, line = 4;
  while(1)
  {
    lcd.setCursor(line, 1);
    if(heure < 10)
    {
      lcd.print(0);
    }
    lcd.print(heure);
    lcd.print(":");
        if(minutes < 10)
    {
      lcd.print(0);
    }
    lcd.print(minutes);
    lcd.print(":");
        if(secondes < 10)
    {
      lcd.print(0);
    }
    lcd.print(secondes);
    delay(1000);
    secondes++;
    if(secondes > 59)
    {
      secondes = 0;
      minutes++;
    }

    if(minutes > 59)
    {
      minutes = 0;
      heure++;
    }

    if(heure > 23)
    {
      minutes = secondes = heure = 0;
    }
```

```
  }
  lcd.clear();
}
```

Code/TP2/TP2.3/TP2.3.ino

2.6 Decompte

De la même façon que nous avons affiché l'heure nous affichons ici un décompte qui part de 9999 jusu'à 0.

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  // set the cursor to (0,0)
  int time = 9999;
  while (time > 0)
  {
    lcd.clear();
    lcd.begin(0,0);
    lcd.print(time);
    delay(1000);
    time--;
  }
  lcd.clear();
}
```

Code/TP2/TP2.4/TP2.4.ino

3 TP3 : Push Push

3.1 Objectif

Dans cette manipulation nous allons voir comment faire intéragir des modules d'entrées avec des modules de sortie pour par exemple, allumer une LED par appuis sur un bouton, afficher le une valeur correspondant à un bouton sur un matrice ou encore faire intéragie une matrice de led avec un joystick. subsectionMatériela

- Un ordinateur
- Un arduino Uno R3
- Un boutton poussoir
- Des resistance de 10KΩ
- Une led
- Une matrice de boutons
- Un afficheur LCD
- Un Joystick
- Une matrice de LED

3.2 LED et boutton

```
const int ledPin = 13; // Utilisation du port 13 pour la led rouge
const int buttonPin = 12; // Utilisation du port 13 pour la led rouge

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, OUTPUT);
}

void loop() {

  buttonState = digitalRead(buttonPin);

  if(buttonState == HIGH)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
```

```
  }
23
}
```

Code/TP3/TP3.1/TP3.1.ino

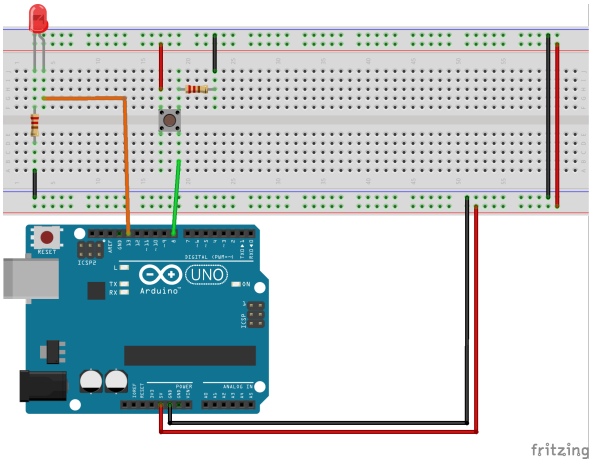


FIGURE 2.6 – LED et boutton

3.3 Debounce

```
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin = 13;       // the number of the LED pin

// Variables will change:
int ledState = HIGH;         // the current state of the output pin
int buttonState;             // the current reading from the input pin
int lastButtonState = LOW;   // the previous reading from the input pin

// the following variables are unsigned long's because the time,
    measured in miliseconds,
// will quickly become a bigger number than can be stored in an int.
unsigned long lastDebounceTime = 0;  // the last time the output pin
    was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if
    the output flickers

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);

  // set initial LED state
  digitalWrite(ledPin, ledState);
}

void loop() {
  // read the state of the switch into a local variable:
  int reading = digitalRead(buttonPin);
```

```
26    // check to see if you just pressed the button
      // (i.e. the input went from LOW to HIGH),  and you've waited
28    // long enough since the last press to ignore any noise:

30    // If the switch changed, due to noise or pressing:
      if (reading != lastButtonState) {
32      // reset the debouncing timer
        lastDebounceTime = millis();
34    }

36    if ((millis() - lastDebounceTime) > debounceDelay) {
        // whatever the reading is at, it's been there for longer
38      // than the debounce delay, so take it as the actual current state:

40      // if the button state has changed:
        if (reading != buttonState) {
42        buttonState = reading;

44        // only toggle the LED if the new button state is HIGH
          if (buttonState == HIGH) {
46          ledState = !ledState;
          }
48      }
      }
50
      // set the LED:
52    digitalWrite(ledPin, ledState);

54    // save the reading.  Next time through the loop,
      // it'll be the lastButtonState:
56    lastButtonState = reading;
  }
```

Code/TP3/TP3.2/TP3.2.ino

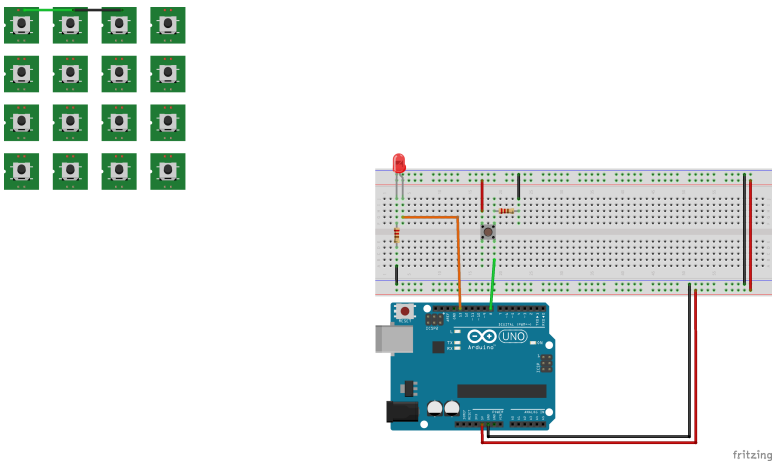


FIGURE 2.7 – Debounce

3.4 Matrice de bouttons

```
1
  /*
3 4x4 matrix keyboard

5 Download the Sketch from my GitHub here.'

7 Its now possible to purchase a 16 switch keyboard for around £2 from
    eBay, an example is here. The 16 keys are arranged in a 4 by 4
    matrix, four rows by four columns. Rows go from side to side (with
    the connector pins at the top) and columns go from top to bottom.
    With the connector at the top key 0 is top left and key 15 is
    bottom right.

9 The sketch shown here has a simple debounce routine, reports the key
    number, then waits for the key to be released. Eight connections
    are required, four for Rows and four for columns. Pins 2,3,4 and 5
    are used for rows and 8,9,10 and 11 are used for columns. On my
    board, with the connections at the top, the left four pins are Rows
     and the right four pins are columns.

11 Once the board is wired to the Arduino upload the sketch below and open
     the Serial monitor, press a button and its number will be shown.
    From top left the keys are 0, 1, 2, 3, the next row down, left to
    right are 4,  5, 6, 7 and so on down to the bottom row which is
    12, 13, 14 and 15.

13 The sketch works as follows, all four Rows are set HIGH, then one Row
    is set LOW. Each column is read and if a key is pressed the
    relevant column will be LOW. Using the Row and Column numbers the
    key number can be calculated. The Rows are reset to HIGH and the
    next Row is set LOW, the columns are scanned again. If no key is
    pressed it will take four Row scans to detect this. The key value
    can be interpreted using a SWITCH loop. Simple switch debounce is
    achieved by inserting a short delay between keyboard scans.

15 /*************************************************
  4x4 keyboard matrix
17 By Chris Rouse Oct. 2015
  Connect:
19
  Row 1 to Arduino pin 2
21 Row 2 to Arduino pin 3
  Row 3 to Arduino pin 4
23 Row 4 to Arduino pin 5
  Column A to Arduino pin 8
25 Column B to Arduino pin 9
  Column C to Arduino pin 10
27 Column D to Arduino pin 11

29 Key identification:
  (with connector at the top)
31 0 1 2 3
  4 5 6 7
33 8 9 10 11
  12 13 14 15
```

```
35
   Connector :
37 Rows      Columns

39 X X X X  X X X X
   1 2 3 4  A B C D
41 (Rows link keys from left to right,
   columns link keys from top to bottom)
43 ************************************************/

45 #include <LiquidCrystal.h>

47 LiquidCrystal lcd(12, 13, 7, 6, A4, A5);

49 int rowCounter =0; // row counter
   int columnCounter =0; // column counter
51 int foundColumn = 0;
   boolean foundCol = false;
53 int keyValue = 0;
   int noKey = 0;
55 boolean readKey = false;
   int debounce = 300; // set this to the lowest value that gives the best
       result
57 const int row1 = 2;
   const int row2 = 3;
59 const int row3 = 4;
   const int row4 = 5;
61 const int colA = 8;
   const int colB = 9;
63 const int colC = 10;
   const int colD = 11;
65 const int ledPin = 13; // onboard LED

67 void setup(){
   Serial.begin(9600);
69 pinMode(row1, OUTPUT);
   pinMode(row2, OUTPUT);
71 pinMode(row3, OUTPUT);
   pinMode(row4, OUTPUT);
73 pinMode(colA, INPUT_PULLUP);
   pinMode(colB, INPUT_PULLUP);
75 pinMode(colC, INPUT_PULLUP);
   pinMode(colD, INPUT_PULLUP);
77 //
   pinMode(ledPin, OUTPUT);
79 digitalWrite(ledPin, LOW); // turn LED off
   }
81
   void loop(){
83 if(noKey == 16){ // no keys were pressed
   readKey = true; // keyboard is ready to accept a new keypress
85 }
   noKey = 0;
87 for(rowCounter=row1; rowCounter<(row4 +1); rowCounter++){
   scanRow(); // switch on one row at a time
89 for(columnCounter = colA; columnCounter <colD +1; columnCounter++){
```

```
readColumn(); // read the switch pressed
if (foundCol== true){
keyValue =(rowCounter-row1) +4*(columnCounter - colA);
}
}
}

if(readKey==true && noKey == 15){ // a key has been pressed
//Serial.println(keyValue); // used for debug
if (keyValue == 13){
//digitalWrite(ledPin, !digitalRead(ledPin)); // toggles LED ON/OFF
}
else{
//digitalWrite(ledPin, LOW);
}

if(readKey)
{
  lcd.clear();
lcd.print(keyValue+1);
}

/*****************************************************
// call to part of the sketch that will use the key number
*/

//*****************************************************
readKey = false; // rest the flag
delay(debounce); // debounce
}
}
void scanRow(){
for(int j =row1; j < (row4 +1); j++){
digitalWrite(j, HIGH);
}
digitalWrite(rowCounter , LOW); // switch on one row
}
void readColumn(){
foundColumn = digitalRead(columnCounter);
if(foundColumn == 0){
foundCol = true;
}
else{
foundCol=false;
noKey=noKey +1; // counter for number of empty columns
}
}
```

Code/TP3/TP3.3/TP3.3.ino

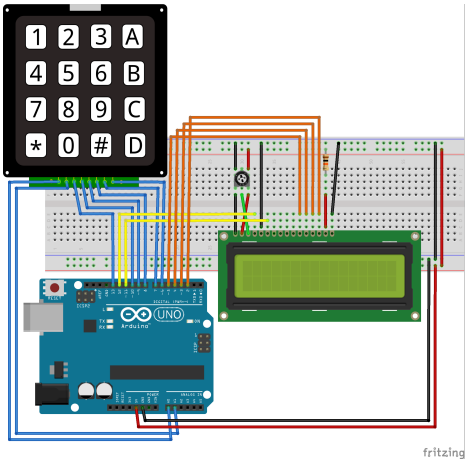


FIGURE 2.8 – Matrice de bouttons

4 TP4 : IR

4.1 Objectif

Dans cette manipulation nous allons voir comment utiliser une télécommande infra rouge et un récepteur infra rouge pour controller des modules à une distance pouvant aller jusqu'a 10 metres. Nous allons donc controller une LED, la couleur d'une LED RGB ainsi qu'un moteur.

4.2 Matériel

- Un ordinateur
- Un arduino Uno R3
- Un récepteur IR
- Des resistance de 10KΩ
- Une led
- Une LED RGB
- Un Un moteur avec controlleur

4.3 Controle d'une LED à distance

```
#include <IRremote.h>
#include <LiquidCrystal.h>

liquidCrystal lcd(12, 11, 5, 4, 3, 2);

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Initialise le recepteur
  lcd.begin{16, 2};
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Recoit la valeur suivante
    switch(results.value)
```

```
22      {
24        case 0xFFA25D:
            Serial.println(" CH-                ");
26          break;
28        case 0xFF629D:
            Serial.println(" CH                 ");
30          break;
32        case 0xFFE21D:
            Serial.println(" CH+                ");
34          break;
36        case 0xFF22DD:
            Serial.println(" PREV               ");
38          break;
40        case 0xFF02FD:
            Serial.println(" NEXT               ");
42          break;
44        case 0xFFC23D:
            Serial.println(" PLAY/PAUSE         ");
46          break;
48        case 0xFFE01F:
            Serial.println(" VOL-               ");
50          break;
52        case 0xFFA857:
            Serial.println(" VOL+               ");
54          break;
56        case 0xFF906F:
            Serial.println(" EQ                 ");
58          break;
60        case 0xFF6897:
            Serial.println(" 0                  ");
62          break;
64        case 0xFF9867:
            Serial.println(" 100+               ");
66          break;
68        case 0xFFB04F:
            Serial.println(" 200+               ");
70          break;
72        case 0xFF30CF:
            Serial.println(" 1                  ");
74          break;
76        case 0xFF18E7:
            Serial.println(" 2                  ");
```

```
78          break;

80        case 0xFF7A85:
            Serial.println("3                ");
82          break;

84        case 0xFF10EF:
            Serial.println("4                ");
86          break;

88        case 0xFF38C7:
            Serial.println("5                ");
90          break;

92        case 0xFF5AA5:
            Serial.println("6                ");
94          break;

96        case 0xFF42BD:
            Serial.println("7                ");
98          break;

100       case 0xFF4AB5:
            Serial.println("8                ");
102         break;

104       case 0xFF52AD:
            Serial.println("9                ");
106         break;

108       default:
            Serial.println("other button     ");
110     }
    }
112 }
```

Code/TP4/TP4.1/TP4.1.ino

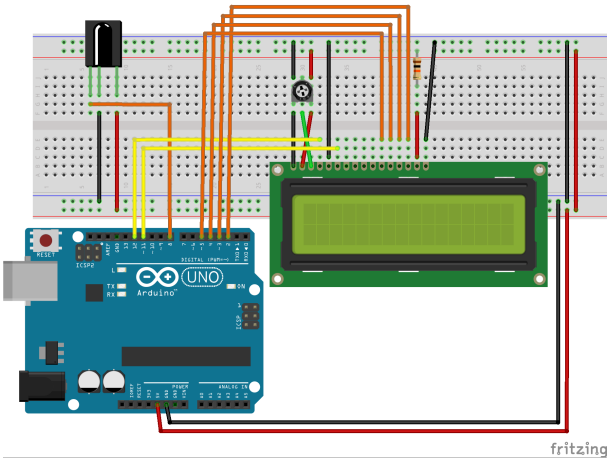


FIGURE 2.9 – Controle d'une LED

4.4 Controle de la couleur d'une LED RGB à distance

```
#include <IRremote.h>

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Initialise le recepteur
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Recoit la valeur suivante
    switch(results.value)
    {
      case 0xFF6897:
        digitalWrite(2, HIGH);
        break;
      case 0xFF30CF:
        digitalWrite(2, LOW);
        break;
    }
  }
}
```

Code/TP4/TP4.2/TP4.2.ino

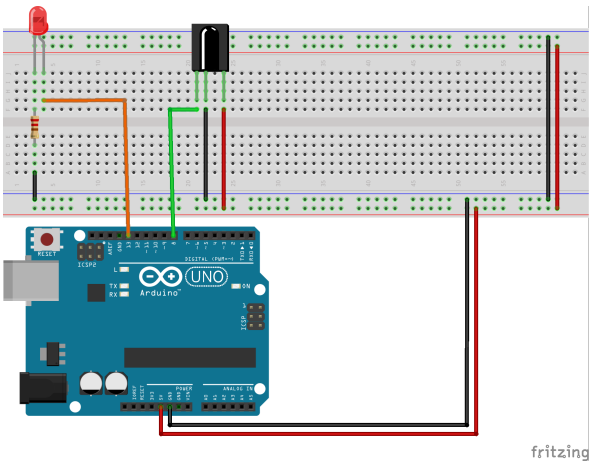


FIGURE 2.10 – Controle d'une LED RGB à distance

4.5 Controle d'un moteur à distance

```
#include <IRremote.h>
```

```
int RECV_PIN = 8;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Initialise le recepteur
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Recoit la valeur suivante
    switch(results.value)
    {
      case 0xFF6897:
        digitalWrite(2, HIGH);
        digitalWrite(3, HIGH);
        digitalWrite(4, HIGH);
        break;
      case 0xFF30CF:
         digitalWrite(2, HIGH);
         digitalWrite(3, LOW);
         digitalWrite(4, LOW);
         break;
      case 0xFF18E7:
        digitalWrite(2, LOW);
        digitalWrite(3, HIGH);
        digitalWrite(4, LOW);
        break;
      case 0xFF7A85:
        digitalWrite(2, LOW);
        digitalWrite(3, LOW);
        digitalWrite(4, HIGH);
        break;
      case 0xFF10EF:
        digitalWrite(2, LOW);
        digitalWrite(3, LOW);
        digitalWrite(4, LOW);
        break;
    }
  }
}
```

Code/TP4/TP4.3/TP4.3.ino

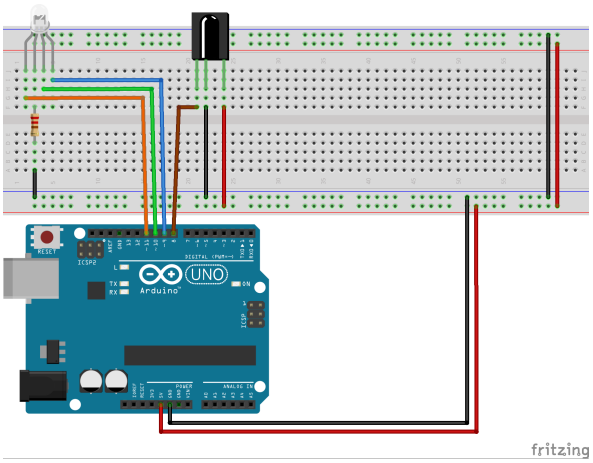


FIGURE 2.11 – Controle d'un moteur à distance

5 TP5 : Motors

5.1 Objectif

Dans cette manipulation nous allons voir comment faire intéragir des modules d'entrées avec des modules de sortie pour par exemple, allumer une LED par appuis sur un bouton, afficher le une valeur correspondant à un bouton sur un matrice ou encore faire intéragie une matrice de led avec un joystick.

5.2 Matériel

a

- Un ordinateur
- Un arduino Uno R3
- Un boutton poussoir
- Des resistance de 10KΩ
- Une led
- Une matrice de boutons
- Un afficheur LCD
- Un Joystick
- Une matrice de LED

5.3 LED et boutton

```
const int ledPin = 13; // Utilisation du port 13 pour la led rouge
const int buttonPin = 12; // Utilisation du port 13 pour la led rouge

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, OUTPUT);
}

void loop() {

  buttonState = digitalRead(buttonPin);

  if(buttonState == HIGH)
```

```
16    {
        digitalWrite(ledPin, HIGH);
18    }
      else
20    {
        digitalWrite(ledPin, LOW);
22    }

24  }
```

Code/TP3/TP3.1/TP3.1.ino

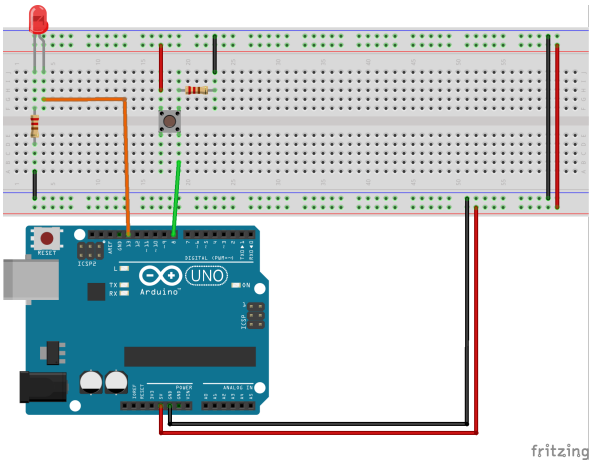


FIGURE 2.12 – LED et boutton

5.4 Debounce

```
const int buttonPin = 2;     // the number of the pushbutton pin
2 const int ledPin = 13;      // the number of the LED pin

4 // Variables will change:
int ledState = HIGH;          // the current state of the output pin
6 int buttonState;            // the current reading from the input pin
int lastButtonState = LOW;    // the previous reading from the input pin
8
// the following variables are unsigned longs because the time,
    measured in miliseconds,
10 // will quickly become a bigger number than can be stored in an int.
unsigned long lastDebounceTime = 0;  // the last time the output pin
    was toggled
12 unsigned long debounceDelay = 50;    // the debounce time; increase if
    the output flickers

14 void setup() {
  pinMode(buttonPin, INPUT);
16  pinMode(ledPin, OUTPUT);

18  // set initial LED state
  digitalWrite(ledPin, ledState);
```

```
}

void loop() {
  // read the state of the switch into a local variable:
  int reading = digitalRead(buttonPin);

  // check to see if you just pressed the button
  // (i.e. the input went from LOW to HIGH), and you've waited
  // long enough since the last press to ignore any noise:

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer
    // than the debounce delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;

      // only toggle the LED if the new button state is HIGH
      if (buttonState == HIGH) {
        ledState = !ledState;
      }
    }
  }

  // set the LED:
  digitalWrite(ledPin, ledState);

  // save the reading. Next time through the loop,
  // it'll be the lastButtonState:
  lastButtonState = reading;
}
```

Code/TP3/TP3.2/TP3.2.ino

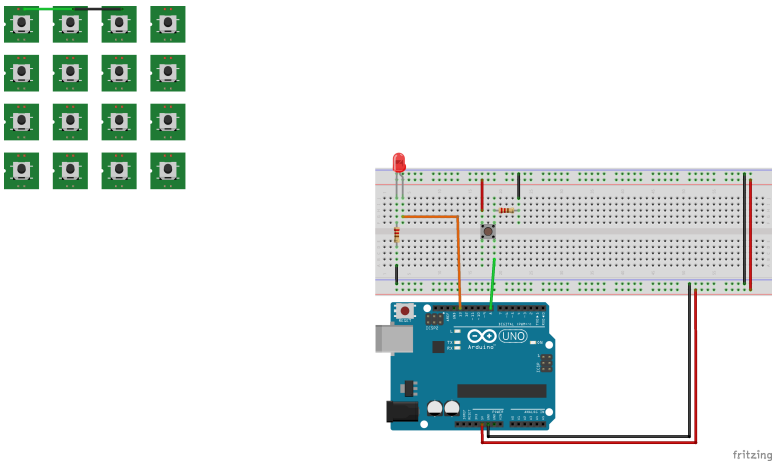


FIGURE 2.13 – Debounce

5.5 Matrice de bouttons

```
1
/*
3 4x4 matrix keyboard

5 Download the Sketch from my GitHub here.'

7 Its now possible to purchase a 16 switch keyboard for around £2 from
    eBay, an example is here. The 16 keys are arranged in a 4 by 4
    matrix, four rows by four columns. Rows go from side to side (with
    the connector pins at the top) and columns go from top to bottom.
    With the connector at the top key 0 is top left and key 15 is
    bottom right.

9 The sketch shown here has a simple debounce routine, reports the key
    number, then waits for the key to be released. Eight connections
    are required, four for Rows and four for columns. Pins 2,3,4 and 5
    are used for rows and 8,9,10 and 11 are used for columns. On my
    board, with the connections at the top, the left four pins are Rows
     and the right four pins are columns.

11 Once the board is wired to the Arduino upload the sketch below and open
     the Serial monitor, press a button and its number will be shown.
    From top left the keys are 0, 1, 2, 3, the next row down, left to
    right are 4, 5, 6, 7 and so on down to the bottom row which is
    12, 13, 14 and 15.

13 The sketch works as follows, all four Rows are set HIGH, then one Row
    is set LOW. Each column is read and if a key is pressed the
    relevant column will be LOW. Using the Row and Column numbers the
    key number can be calculated. The Rows are reset to HIGH and the
    next Row is set LOW, the columns are scanned again. If no key is
    pressed it will take four Row scans to detect this. The key value
    can be interpreted using a SWITCH loop. Simple switch debounce is
    achieved by inserting a short delay between keyboard scans.
```

```
/*************************************************
4x4 keyboard matrix
By Chris Rouse Oct. 2015
Connect:

Row 1 to Arduino pin 2
Row 2 to Arduino pin 3
Row 3 to Arduino pin 4
Row 4 to Arduino pin 5
Column A to Arduino pin 8
Column B to Arduino pin 9
Column C to Arduino pin 10
Column D to Arduino pin 11

Key identification:
(with connector at the top)
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15

Connector:
Rows      Columns

X X X X  X X X X
1 2 3 4  A B C D
(Rows link keys from left to right,
columns link keys from top to bottom)
*************************************************/

#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 13, 7, 6, A4, A5);

int rowCounter =0; // row counter
int columnCounter =0; // column counter
int foundColumn = 0;
boolean foundCol = false;
int keyValue = 0;
int noKey = 0;
boolean readKey = false;
int debounce = 300; // set this to the lowest value that gives the best
    result
const int row1 = 2;
const int row2 = 3;
const int row3 = 4;
const int row4 = 5;
const int colA = 8;
const int colB = 9;
const int colC = 10;
const int colD = 11;
const int ledPin = 13; // onboard LED

void setup(){
Serial.begin(9600);
pinMode(row1, OUTPUT);
```

```
pinMode(row2, OUTPUT);
pinMode(row3, OUTPUT);
pinMode(row4, OUTPUT);
pinMode(colA, INPUT_PULLUP);
pinMode(colB, INPUT_PULLUP);
pinMode(colC, INPUT_PULLUP);
pinMode(colD, INPUT_PULLUP);
//
pinMode(ledPin, OUTPUT);
digitalWrite(ledPin, LOW); // turn LED off
}

void loop(){
if(noKey == 16){ // no keys were pressed
readKey = true; // keyboard is ready to accept a new keypress
}
noKey = 0;
for(rowCounter=row1; rowCounter<(row4 +1); rowCounter++){
scanRow(); // switch on one row at a time
for(columnCounter = colA; columnCounter <colD +1; columnCounter++){
readColumn(); // read the switch pressed
if (foundCol== true){
keyValue =(rowCounter-row1) +4*(columnCounter - colA);
}
}
}

if(readKey==true && noKey == 15){ // a key has been pressed
//Serial.println(keyValue); // used for debug
if (keyValue == 13){
//digitalWrite(ledPin, !digitalRead(ledPin)); // toggles LED ON/OFF
}
else{
//digitalWrite(ledPin, LOW);
}

if(readKey)
{
  lcd.clear();
lcd.print(keyValue+1);
}

/*****************************************************
// call to part of the sketch that will use the key number
*/

//*****************************************************
readKey = false; // rest the flag
delay(debounce); // debounce
}
}
void scanRow(){
for(int j =row1; j < (row4 +1); j++){
digitalWrite(j, HIGH);
}
digitalWrite(rowCounter , LOW); // switch on one row
```

```
}
void readColumn(){
foundColumn = digitalRead(columnCounter);
if(foundColumn == 0){
foundCol = true;
}
else{
foundCol=false;
noKey=noKey +1; // counter for number of empty columns
}
}
```

Code/TP3/TP3.3/TP3.3.ino

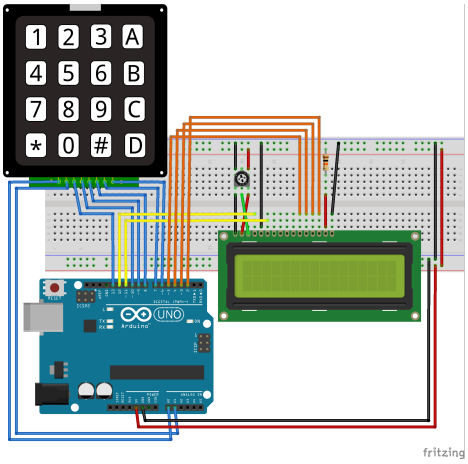


FIGURE 2.14 – Matrice de bouttons

6 Remerciement

Je remercie Terencio Agozzino pour avoir réalisé la mise en page de ce document en LaTeX.