|  |  |  |
| --- | --- | --- |
|  | Expérience de lévitation | 02/06/2020 |

Le but de notre montage est de faire léviter des objets de petites tailles, par exemple des grains de polystyrène ou des gouttes d’eau.

**Matériel (Seulement les composants, les consommables ne sont pas spécifiés) :**

• 12 transducteurs Ultrasons (KPUS-40T-16T-K768)  
 • 1 Arduino Uno  
 • 1 driver pour moteur (L298)

**Réalisation de la structure :**

|  |  |
| --- | --- |
|  | La structure du système a été imprimée avec une imprimante 3D. Il est constitué de deux socles de forme hexagonale où seront disposer les transducteurs. Les deux socles sont reliés par l’intermédiaire de trois bras. La matière utilisée est le PLA afin de minimiser les coûts et faciliter la fabrication du prototype. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Figure 1 : modelisation du socle* | *Figure 2 : assemblage numérique de la structure* | *Figure 3 : Assemblage de la structure avec les transducteurs* |

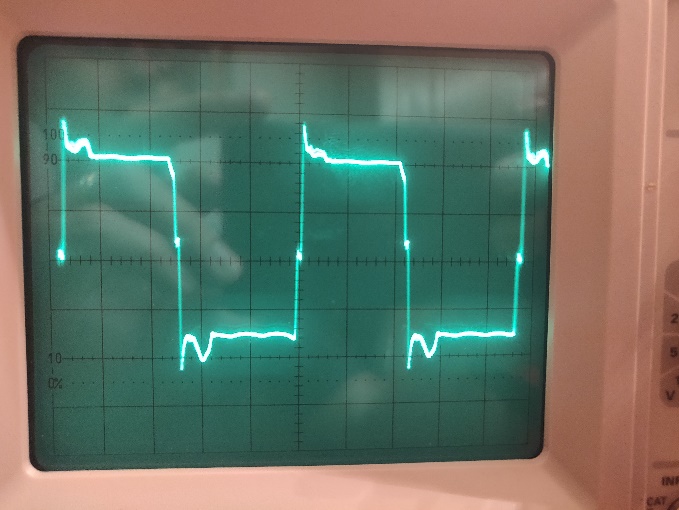
**Réalisation du circuit :**

|  |  |
| --- | --- |
|  | Le principe est d’alimenter les transducteurs avec une tensions alternatives . Pour se faire, il suffit d’utiliser la modulation par largeur d’impulsion de l’Arduino afin de générer cette tension. Cependant, l’Arduino ne peut fournir une tension maximum . Cette tension n’est pas suffisante et il faut amplifier le signal avec le driver et une alimentation externe continue de .   Il faut ensuite souder les transducteurs en dérivation puis les relier au driver.  Le schéma du circuit est en annexe *(figure 5)* |
| *Figure 4 : Montage experimental* |  |

**Premier Résultat :**

|  |  |
| --- | --- |
|  | L’expérience n’est actuellement pas concluante. Seulement, nous n’avons pas les moyens matériels de vérifier notre circuit pour le moment, notamment la nature et les caractéristiques (fréquence, amplitude et offset principalement) du signal fournis par l’Arduino afin de corriger les éventuels défauts. De plus le code *(figure 6)* récupéré de la thèse **TinyLev** *(Bibliographie n°1)* est très complexe et nous travaillons actuellement dessus afin de mieux le maîtriser. |

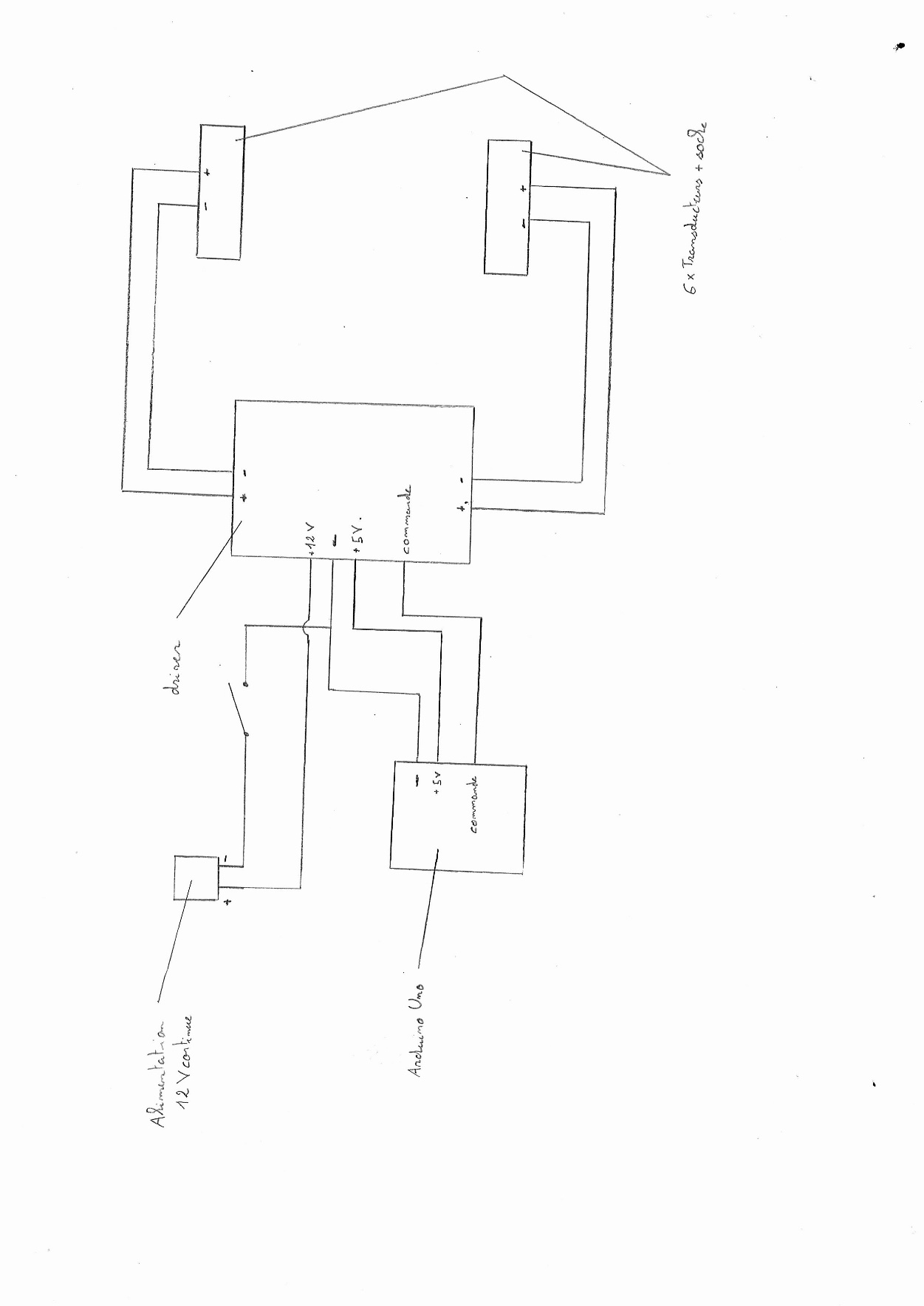
**Retour sur la première expérience :**



*Figure 5 : Photo de l’ecran de l’oscilloscope.  
Calibre vertical : 5V/div   
Calibre horizontal : 2μs/div*

|  |  |
| --- | --- |
|  | On relève alors une tension crête à crête ce qui est proche de ce qui était attendu. Cependant, on relève une période .  La fréquence est donc . La fréquence est alors beaucoup trop élevée par rapport à la fréquence théorique  Nous pouvons essayer de remplacer l’Arduino Uno par un générateur basse fréquence pour générer un signal plus proche de la fréquence théorique cible. |

**Annexe :**



*Figure 6 : Schéma du circuit électrique*

|  |
| --- |
| #include <avr/sleep.h>  #include <avr/power.h>  #define N\_PORTS 1  #define N\_DIVS 24  #define WAIT\_LOT(a) \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop");\_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop");\_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop");\_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop")  #define WAIT\_MID(a) \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop");\_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop");\_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop")  #define WAIT\_LIT(a) \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop"); \_\_asm\_\_ \_\_volatile\_\_ ("nop")  #define OUTPUT\_WAVE(pointer, d) PORTC = pointer[d\*N\_PORTS + 0]  #define N\_BUTTONS 6  #define STEP\_SIZE 1  #define BUTTON\_SENS 2500  #define N\_FRAMES 24  static byte frame **=** 0**;**  static byte animation**[**N\_FRAMES**][**N\_DIVS**]** **=**  **{{**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x6**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x6**,**0x6**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x6**,**0x6**,**0x6**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x6**,**0x6**,**0x6**,**0x6**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x5**,**0x5**,**0x5**,**0x5**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0xa**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x5**,**0x5**,**0x5**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0xa**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x5**,**0x5**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0xa**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x5**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0xa**},**  **{**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0xa**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0xa**,**0xa**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0xa**,**0xa**,**0xa**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x5**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0xa**,**0xa**,**0xa**,**0xa**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x9**,**0x9**,**0x9**,**0x9**,**0x9**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0x6**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x9**,**0x9**,**0x9**,**0x9**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0x6**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x9**,**0x9**,**0x9**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0x6**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x9**,**0x9**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0x6**,**0x6**},**  **{**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x5**,**0x9**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0xa**,**0x6**}};**  void setup**()**  **{**  DDRC **=** 0b00001111**;** //A0 to A3 are the signal outputs  PORTC **=** 0b00000000**;**    pinMode**(**10**,** OUTPUT**);** //pin 10 (B2) will generate a 40kHz signal to sync  pinMode**(**11**,** INPUT\_PULLUP**);** //pin 11 (B3) is the sync in  **for** **(**int i **=** 2**;** i **<** 8**;** **++**i**){** //pin 2 to 7 (D2 to D7) are inputs for the buttons  pinMode**(**i**,** INPUT\_PULLUP**);**  **}**  // generate a sync signal of 40khz in pin 10  noInterrupts**();** // disable all interrupts  TCCR1A **=** bit **(**WGM10**)** **|** bit **(**WGM11**)** **|** bit **(**COM1B1**);** // fast PWM, clear OC1B on compare  TCCR1B **=** bit **(**WGM12**)** **|** bit **(**WGM13**)** **|** bit **(**CS10**);** // fast PWM, no prescaler  OCR1A **=** **(**F\_CPU **/** 40000L**)** **-** 1**;**  OCR1B **=** **(**F\_CPU **/** 40000L**)** **/** 2**;**  interrupts**();** // enable all interrupts  // disable everything that we do not need  ADCSRA **=** 0**;** // ADC  power\_adc\_disable **();**  power\_spi\_disable**();**  power\_twi\_disable**();**  power\_timer0\_disable**();**  Serial**.**begin**(**115200**);**  byte**\*** emittingPointer **=** **&**animation**[**frame**][**0**];**  byte buttonsPort **=** 0**;**  bool anyButtonPressed**;**  bool buttonPressed**[**N\_BUTTONS**];**  short buttonCounter **=** 0**;**  LOOP**:**  **while(**PINB **&** 0b00001000**);** //wait for pin 11 (B3) to go low    OUTPUT\_WAVE**(**emittingPointer**,** 0**);** buttonsPort **=** PIND**;** WAIT\_LIT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 1**);** anyButtonPressed **=** **(**buttonsPort **&** 0b11111100**)** **!=** 0b11111100**;** WAIT\_MID**();**  OUTPUT\_WAVE**(**emittingPointer**,** 2**);** buttonPressed**[**0**]** **=** buttonsPort **&** 0b00000100**;** WAIT\_MID**();**  OUTPUT\_WAVE**(**emittingPointer**,** 3**);** buttonPressed**[**1**]** **=** buttonsPort **&** 0b00001000**;** WAIT\_MID**();**  OUTPUT\_WAVE**(**emittingPointer**,** 4**);** buttonPressed**[**2**]** **=** buttonsPort **&** 0b00010000**;** WAIT\_MID**();**  OUTPUT\_WAVE**(**emittingPointer**,** 5**);** buttonPressed**[**3**]** **=** buttonsPort **&** 0b00100000**;** WAIT\_MID**();**  OUTPUT\_WAVE**(**emittingPointer**,** 6**);** buttonPressed**[**4**]** **=** buttonsPort **&** 0b01000000**;** WAIT\_MID**();**  OUTPUT\_WAVE**(**emittingPointer**,** 7**);** buttonPressed**[**5**]** **=** buttonsPort **&** 0b10000000**;** WAIT\_MID**();**  OUTPUT\_WAVE**(**emittingPointer**,** 8**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 9**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 10**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 11**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 12**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 13**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 14**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 15**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 16**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 17**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 18**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 19**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 20**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 21**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 22**);** WAIT\_LOT**();**  OUTPUT\_WAVE**(**emittingPointer**,** 23**);**  **goto** LOOP**;**  **}**  void loop**(){}** |

*Figure 7 : Code en C pour l’Arduino*