Description of Code

Student name: Yifan Feng

Student QMUL email: y.feng@se20.qmul.ac.uk

Date: 18 Oct 2020

General Purpose

The three sketches together aim to make the snake game more interactive regarding the use of physical components (e.g. buttons, motors). The main concept is that players use a simplified control panel to steer the snake's movement (i.e. left, right, up and down) and adjust its moving speed (i.e. fast or slow) by a potentiometer. Motor is used to demonstrate boundary collision (i.e. game over).

How the Sketch Works

This project mainly relies on the Arduino example projects as well as tutorial videos about potentiometer, motor events and the serial communications. The main resources are:

- https://www.adeept.com/learn/tutorial-41.html
- https://www.bilibili.com/video/BV1YW411Z76E
- https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing/all#shaking-hands-part-1

Sketch One:

The first step is to re-edit Processing codes and connect Processing to Arduino via serial communications. The design guideline is to send different serial signals, which connotes four directions to Processing. As for Processing, the important variable is *int val*, which is used to receive serial data from Arduino. Later *val* will read and store new data update from different button signals (e.g. 1 refers to left). Snake direction is controlled by four buttons and each button is assigned a unique reference number. The new function named *Void ked(int a)* uses Boolean (==) to detect input value and accordingly responds. For example, if pressing the button 1, a signal value 1 is transmitted to Processing and matched up with the key case "L", the snake will turn left. As for Arduino, the important variable is *int switchPin*. I used *int* to set up input pin. SwitchPin 1 to 4 connects to four buttons on the breadboard and by reading input (button pressed = HIGH = on), each button will give an output (e.g. *Serial.write(value)*).

Sketch Two:

The second step is to use a potentiometer to speed up or slow down the running snake. In Arduino, I created a new integer value named *int SensorValue* (by default is zero). Its usage is to distinguish the value from *Serial.write( )* for switching directions. As stated, digits 1 to 4 represent four moving directions respectively. Digit 5 to maximum sensor value (by default is 1023) links to the speed change. In Processing, I kept using *int val* to receive and store input value from Arduino. To detect the function (i.e. direction switch or speed change) of input value, I used an if statement: if the input digit is less than five, *Ked(val)* function will shift the direction; if the input digit is greater than five but less than max sensor value 1023, *Speed* function will alter the current pace. Additionally, I added a *println(val)* to check on the input value (for debugging).

Sketch Three:

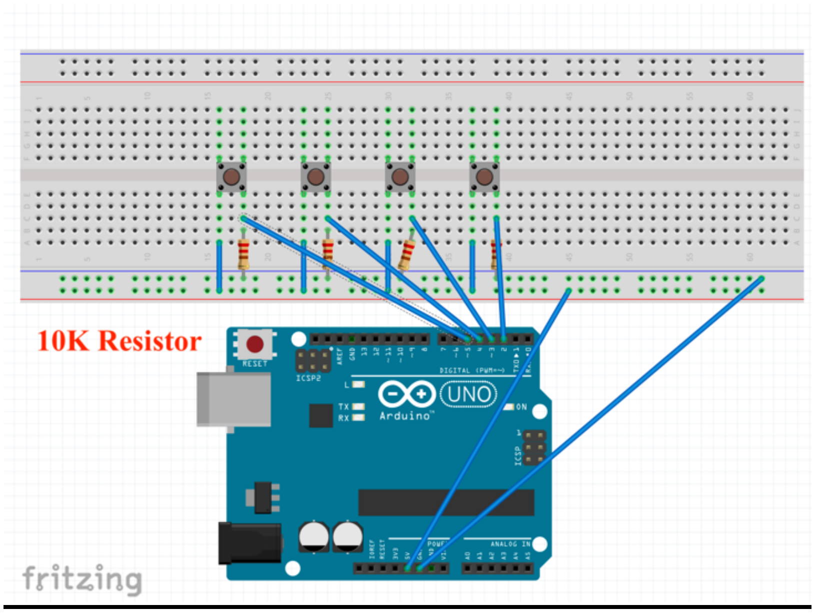
The third sketch aims to add a motor spinning event when boundary collision happens. The coding schedule is when game over happens, Processing sends a signal to Arduino to trigger motor. In Processing *void gameover( )*, I added a new line *myPort.write(0)* to export 0 when the snake hitting the boundary (aka game over). In Arduino, *void serialEvent( )* reads the input signal 0 and activates the new loop about spinning time. The connection is *Boolean value start.* By default, *bool start* is false. I created a new integer called *int time* to count passing time. If time is less than 300, motor is powered and starts working. If it is greater than 300, timer will be reset to 0 and stop spinning.

Reflections and Improvements

In general, I found that building an Arduino project is easier than it in Processing. There are example sketches for further modification in each case. The difficulty is found in using serial signals to communicate two platforms. In my project, I did not attach string name (string A, B, C) to distinguish different use of serial digits. I tried to follow the tutorial instructions but could not manage to do it. Therefore, to be easier, I only applied serial signals (0-1023) and divided them into three parts (i.e. digit 1 to 4 connotes directions, 5 to 1023 connotes speed, and 0 is used to detect game over). As for improvement, I should have added string characters to read each signal's purpose (e.g. A1 to switch direction and B1 to modify speed). I found my projects has a significantly delayed response between two platforms even I had increased the speed of serial communication (from 9600 to 19800). It takes a while for Processing to receive button press signals and gives feedback. When *game over* is true, motor will spin after noticeable response time. Also, due to the flaws in my original Processing file – the snake game, for example, when I adjust the speed via potentiometer, the snake structure will be affected accordingly and result in broken pieces (not a line anymore).

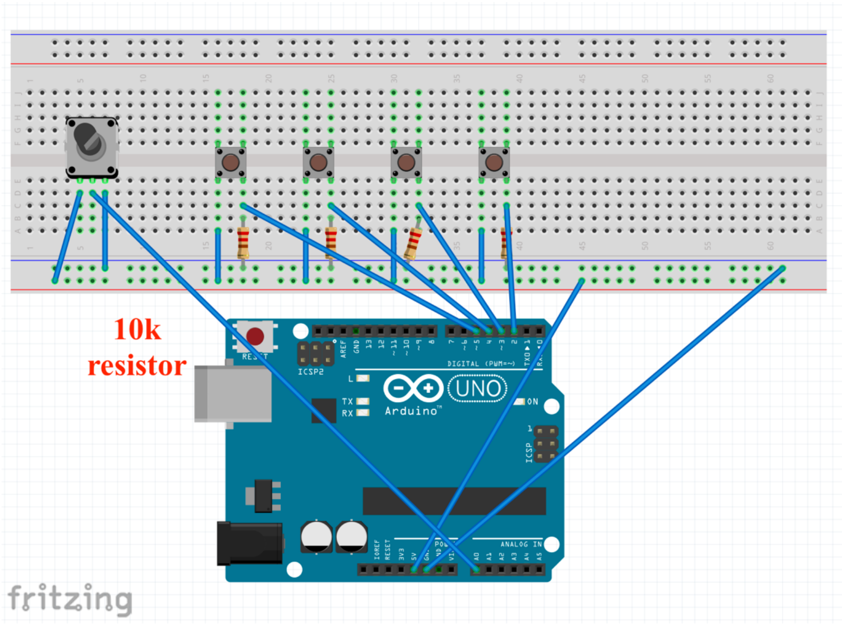
Appendix: Circuit Diagram

N.B. In the picture, the motor and resistor are different from Arduino Uno set. Attached photo is my final work.



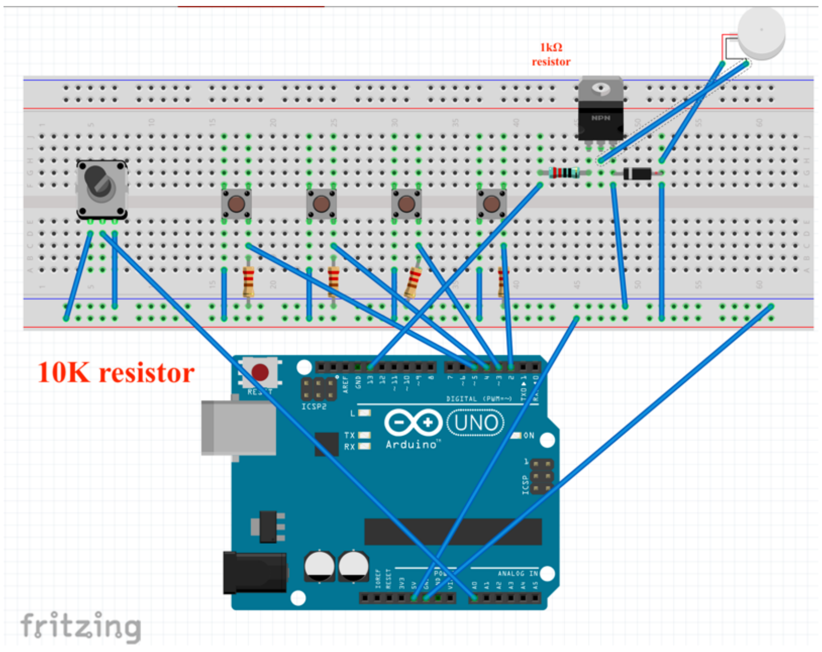
(sketch one): button pin 2,3,4,5

- For button events, I used 10K resistor for each button. Pin number2,3,4,5 connects to LEFT, RIGHT, DOWN, UP.



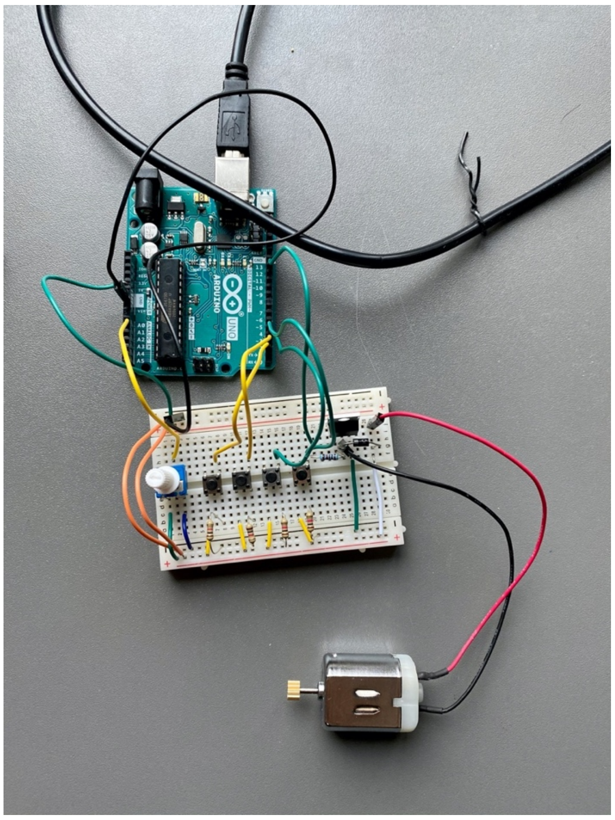
(sketch two): button pin:2,3,4,5, potentiometer A0

- A0 is the potentiometer input pin.



(sketch three): button pin 2,3,4,5, potentiometer A0, motor pin 13

- Motor input pin is 13.
- I used a 1K resistor and a MOSFET and diode polarity



(my project)