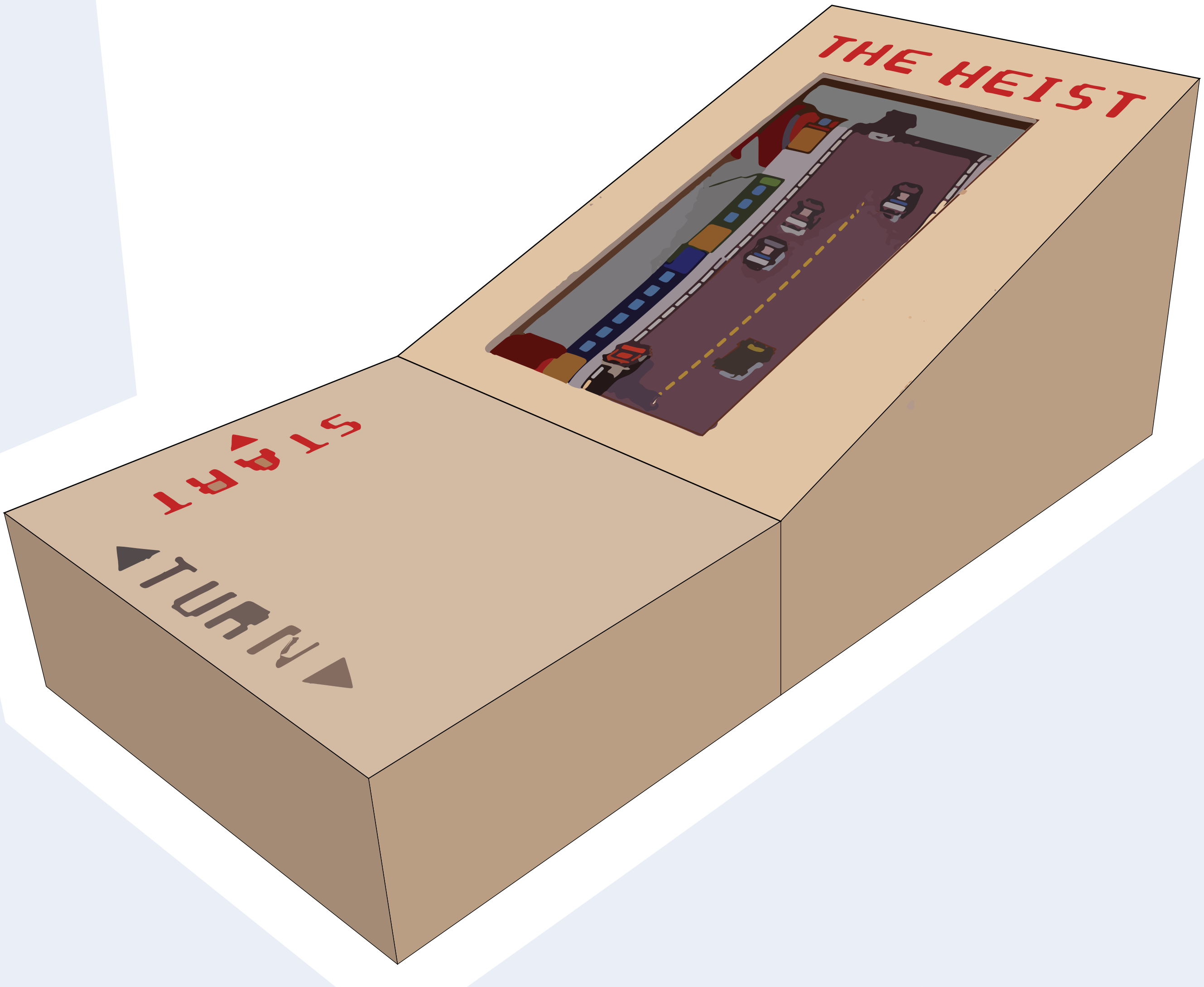
PHYSICAL VIDEOGAME

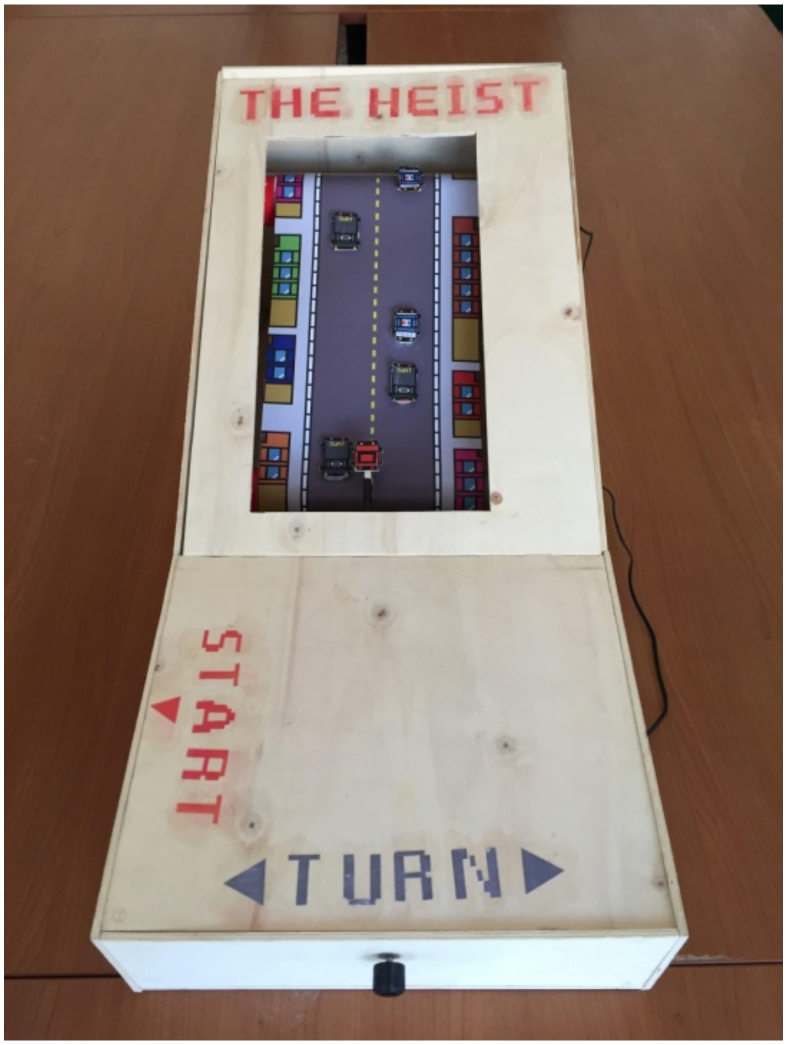


LINK CAMPUS UNIVERSITY

DASIC
DIGITAL ADMINISTRATION AND SOCIAL INNOVATION CENTRE

intel

HOW TO BUILD A PHYSICAL VIDEOGAME WITH INTEL GALILEO



Tutorial by

Antongiulio Calabrìa
antocalab@gmail.com

Valerio De Cecio
valeriodececio@gmail.com

Roberta Grimaldi
robertagrim@gmail.com

The project "Physical Videogame" is realized under the supervision of Carlo Maria Medaglia and Massimiliano Dibitonto, within the ***Digital Administration and Social Innovation Center (Dasic Lab), Link Campus University.***

Below a very simple tutorial to recreate one of the many car videogames of the past (like Death Race, Pole Position, Formula One) using the Intel Galileo Board (first generation).
The user can play moving own avatar around barriers positioned along the path. The avatar is moved through a servo motor 0-180° oriented by a potentiometer knob. Videogame background scroll on a reel, activated by a continuous rotation servo motor. When the avatar crashes one barrier, the system reveals the clash thanks by magnetic sensor arranged under barriers, so it stops the game.
This project has also been selected and showcased in the Intel booth at the Maker Faire Rome 2014.

Required materials (each):

- 1 continuous servo-motor
- 1 0-180 servo-motor
- 1 push button
- 1 breadboard
- 1 galileo board (1st generation)
- 1 hall effect sensor
- 1 potentiometer knob 10k ohm
- some magnets
- some jumper wires
- 1 pvc paper sheet (on which is printed the road)
- 2 pipes (to make a reel)
- 4 holder (to bear the pipes)
- some printed cars
- some plywood sheets (to build the box)

Recommended age

15+

Recommended age (to play with)

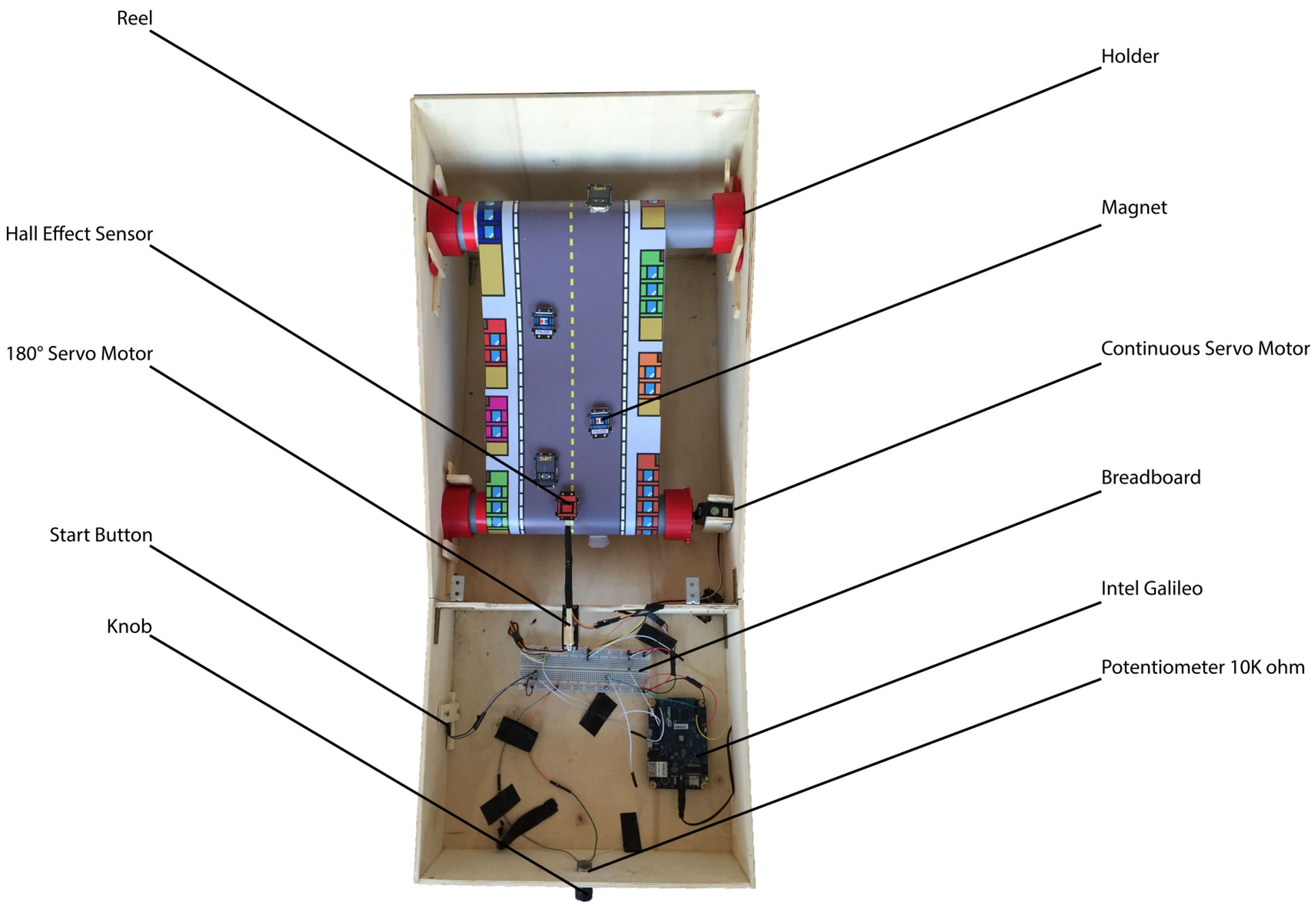
3+ :)

Minimum setup time

1 days

Video (italian)

https://www.youtube.com/watch?v=NCO82FdSn_w



Summary
Step 1 - Build the plywood box
Step 2 - Print or buy holders/cut pipes/print road&cars
Step 3 - Connect and test continuous servo-motor
Step 4 - Connect and test 0-180 servo-motor
Step 5 - Connect and test hall effect sensor
Step 6 - Connect and test the potentiometer
Step 7 - Connect, test and start the game with a button
Step 8 - Full code & Fritzing

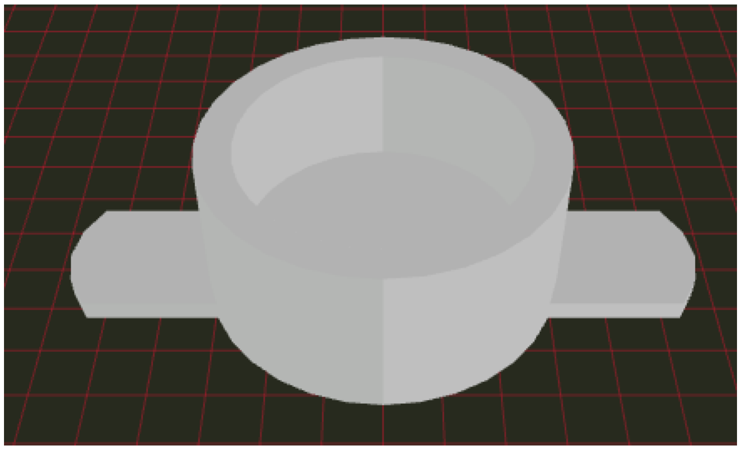
Step 1 - Build the plywood box

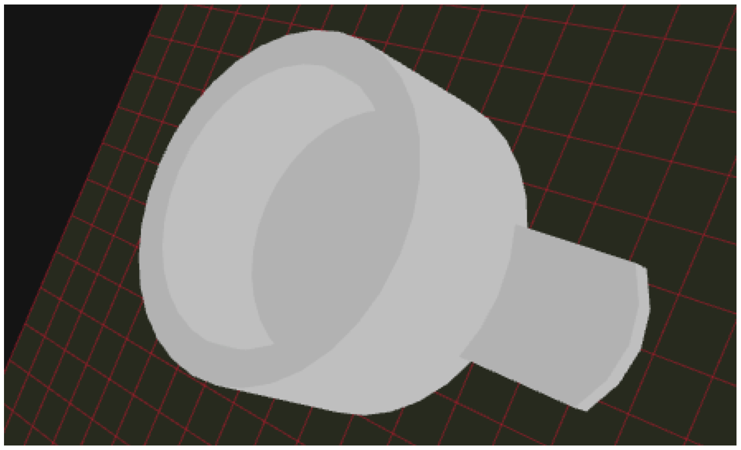
In our example the box containing the game was 70 x 30 x 30 cm, we used 11 pieces of plywood to build a retro-gaming style box. The box is assembled with wood glue and small nails.

Step 2 - Print or buy holders / cut pipes / print road&cars

We used two hydraulic pipes as reels. We cut them based on the width of the box.

We printed 4 reels holders to hold the pipes (images below). This link allows the download of .stl file for the 3D printing of these pieces.

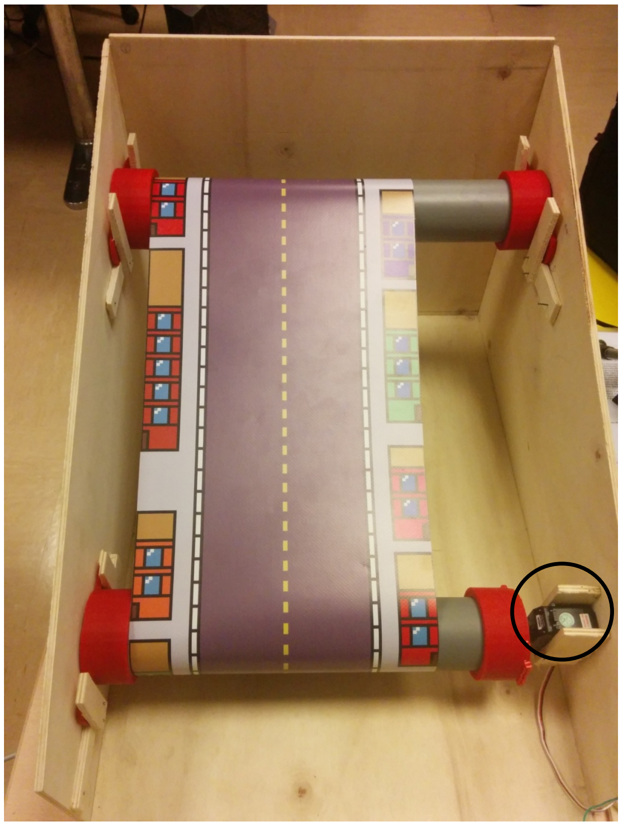




We decide to print the street on a pvc paper sheet for ensure a better traction on the reels, allowing a continuous rotation of the street. This link allows the download of .ai file of the street.
We printed various police cars, SWAT vans and simple cars. This link allows the download of .ai file of the cars.

Step 3 - Connect and test continuous servo-motor

The continuous servo-motor is glued to an holder which is the only one fixed to a pipe and is the one in charge to turn the roll.



To connect the servo to the board you have to:

- connect the red cable to 5V pin
- connect the black cable to ground
- connect the yellow cable to a pwm pin

Once the servo is connected you can test it running some example code which is pasted below.

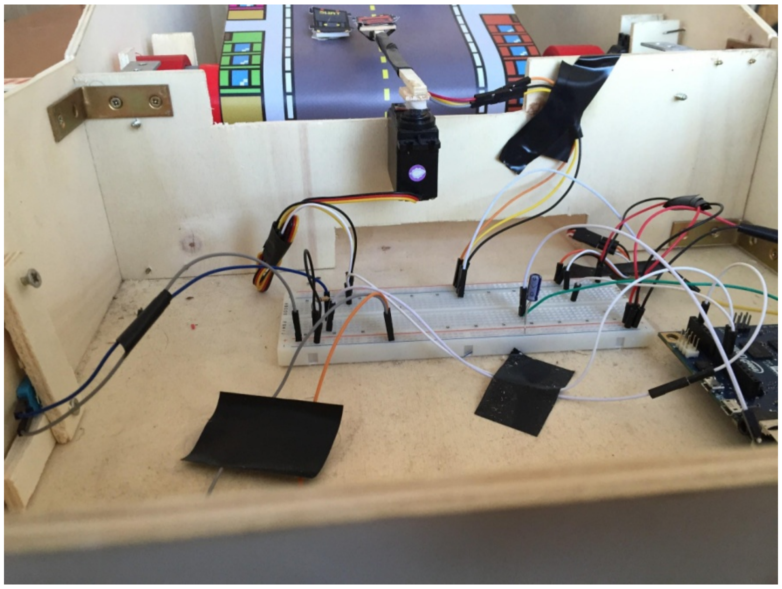
```
Servo myservo;

void setup()
{
  myservo.attach(9);
  myservo.write(180);  // move the servo full speed in a direction
}

void loop() {}
```

On a continuous rotation servo, this will set the speed of the servo (with 0 being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement).

You can alternate the rotation direction changing the value of `myservo.write(180)` to `myservo.write(0)`.

Step 4 - Connect and test 0-180 servo-motor

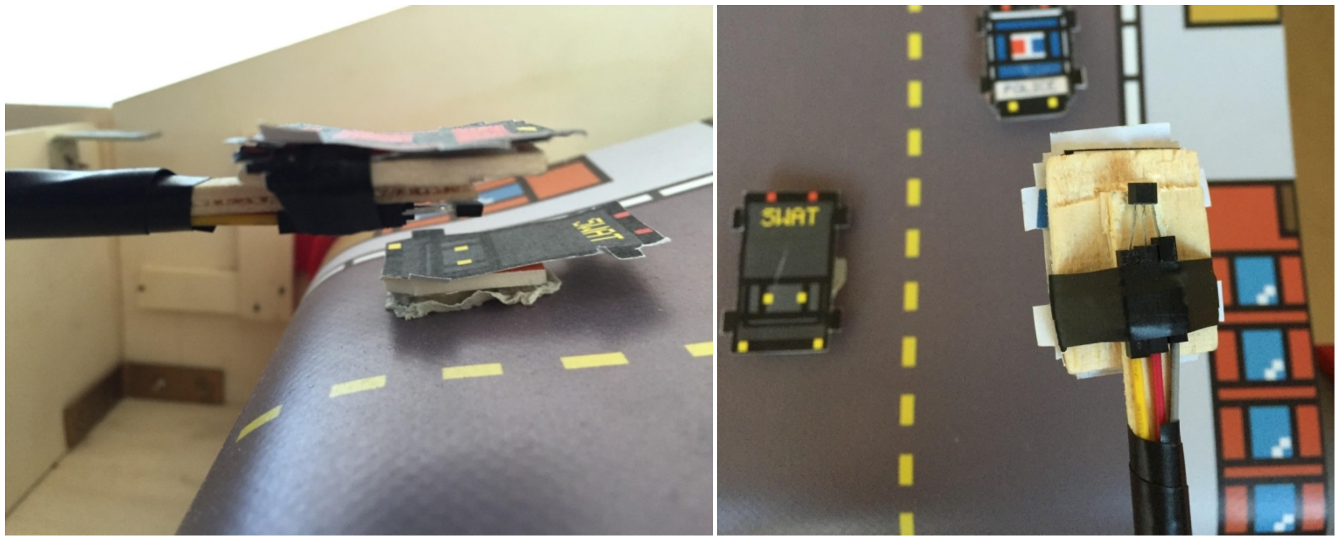
This servo moves the arm with the player's car although we will not use the full rotational arc but will set a threshold to stop the arm movement.



The connection is the same as before but this time we will control the servo with a knob, which will be explained in the step 6.

Step 5 - Connect and test hall effect sensor

The hall effect sensor can detect the magnetic field if it's directly under itself. We use this product of Sparkfun https://www.sparkfun.com/products/9312.
We will attach this sensor under the player's car and we will emulate collisions with other cars when this small sensor detect the magnet under the other cars.

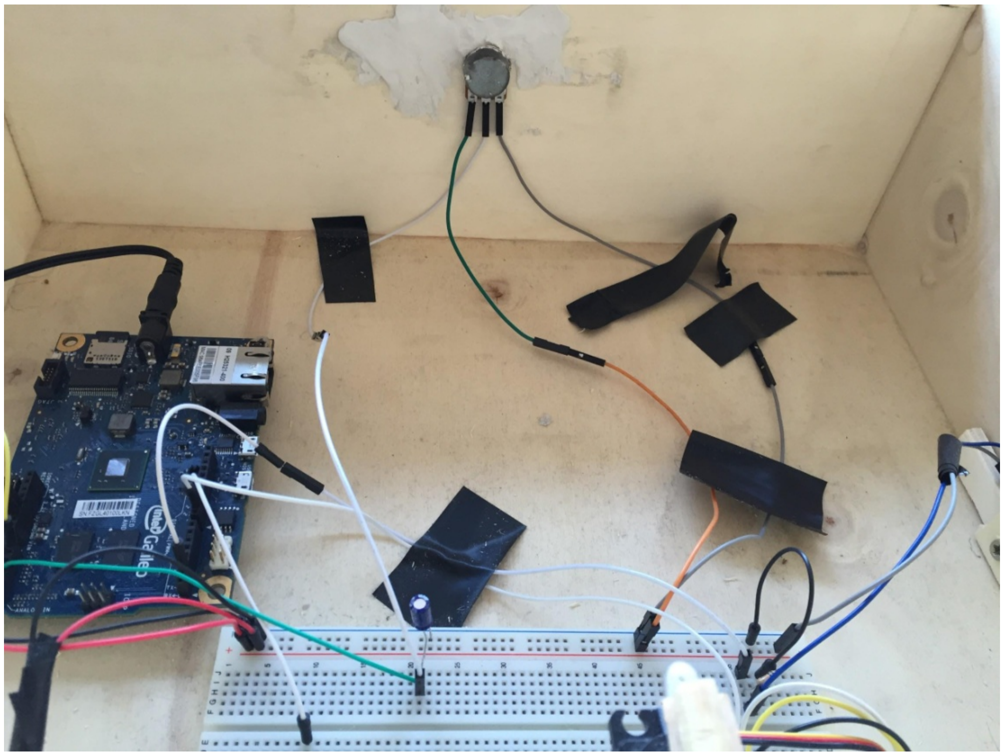


Connection is usual, one pin goes to an analog port, number 0 in our code, plus it has positive and negative pins.
The following code shows how we interact with the value returned from the sensor, we will do a breakthrough into the code in step 8.

```
void DoMeasurement(){
// measure magnetic field

  int raw = analogRead(0);   // Range : 0..1024


  long compensated = raw - NOFIELD;                 // adjust relative to no applied field
  long gauss = compensated * TOMILLIGAUSS / 1000;   // adjust scale to Gauss



  streetServo.write(0); //move the road

  gauss = abs(gauss);

  Serial.print(gauss);
  Serial.print(" Gauss ");
  Serial.println();

  if( gauss < 100){
    streetServo.write(0); //if we do not collide with a magnet we keep moving
  } else {
    collision(); //otherwise we manage a collision
  }
}

void collision(){
  streetServo.write(91); //stop the road
  start = false; //we move to STOPPED state
}
```

Step 6 - Connect and test the potentiometer

The potentiometer will move the player to the left or to the right. As usual we will connect the red wire to power and the black wire to the ground. The white wire will be connected to the analog pin of the board and the board will map the electric potential to the player's position.



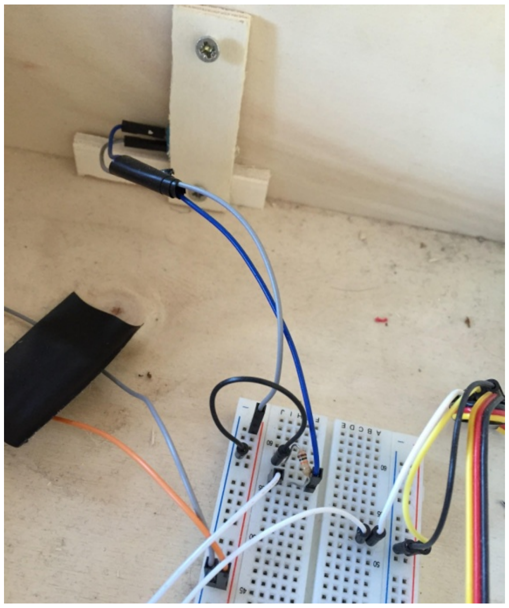
```
void movePlayer(){
  val = analogRead(3);             // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 179);  // scale it to use it with the servo (value between 0 and 180)
  if((val - oldVal) < -5){
    isRight = false;
    moveServo(50);
  }else if(val - oldVal > 5){
    isRight = true;
    moveServo(65);
  }
  oldVal = val;
}

void moveServo(int n){
  playerServo.write(n);
}
```

Step 7 - Connect, test and start the game with a button

The button will allow to start the game. To connect the button you will have to:

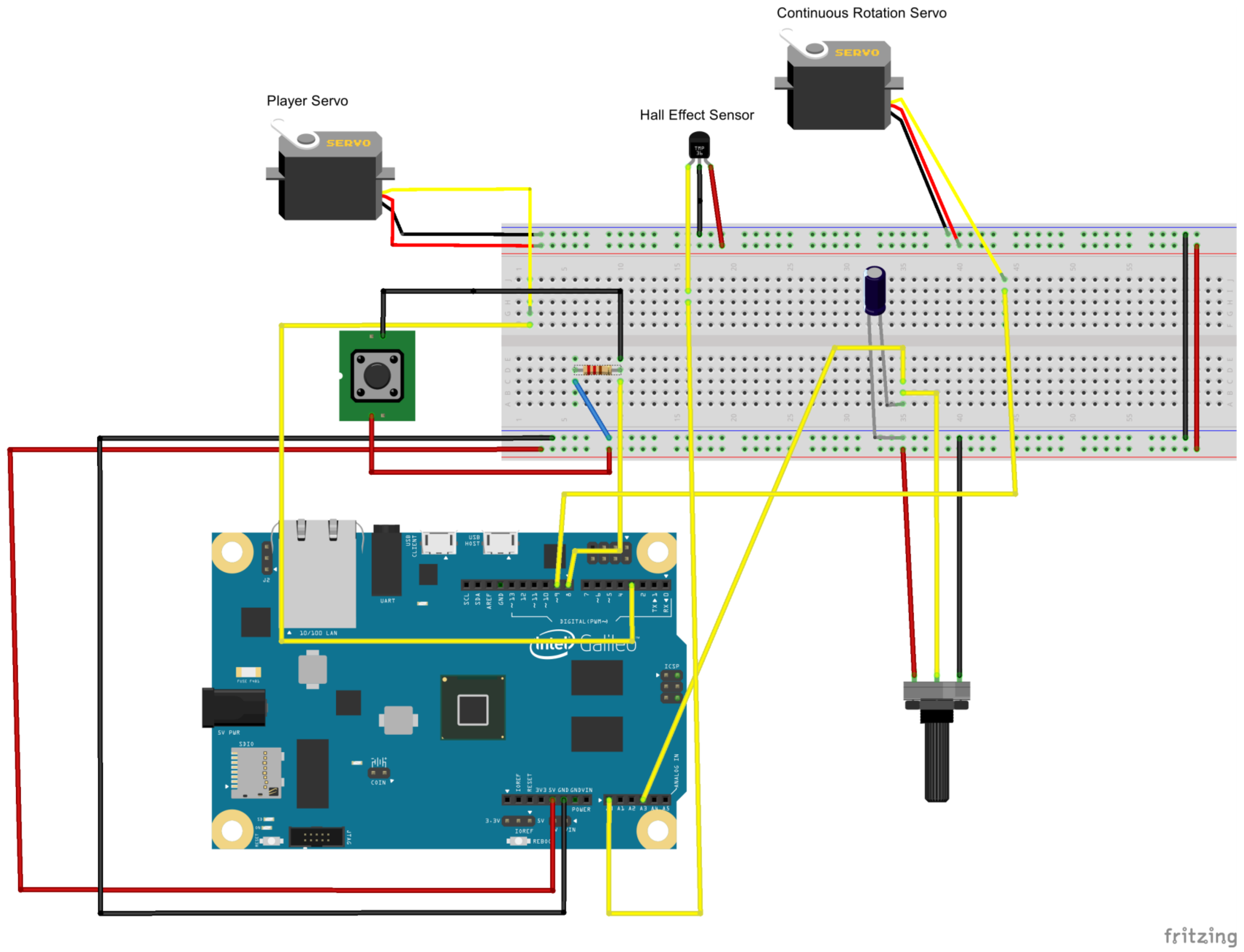
- connect the red cable to 5V
- connect the black cable, through a 10k ohm resistor, to ground
- through the black cable, connect the yellow cable to the board logic pin to pass the signal.

We printed a 3D button and stuck it on the button connected to the board.



For the code see the step 8.

Full code & Fritzing



```
#include <Servo.h>

#define NOFIELD 505L
#define TOMILLIGAUSS 1953L

Servo playerServo, streetServo;
int val, oldVal=55;;

int buttonPin = 8;
int buttonState, lastButtonState = 0;

int shroom; //Immunity variable

boolean start, is Right;


void setup() {
Serial.begin(9600);
playerServo.attach(3);
playerServo.set48hz();
streetServo.attach(9);
streetServo.set48hz();
start = true;
pinMode(buttonPin, INPUT);
shroom = 0;
}

void DoMeasurement() { //function to check if player carashed with a magnet

// measure magnetic field
int raw = analogRead(0);   // Range : 0..1024


long compensated = raw - NOFIELD;                 // adjust relative to no applied field
long gauss = compensated * TOMILLIGAUSS / 1000;   // adjust scale to Gauss



streetServo.write(0); //start moving the road

gauss = abs(gauss);

Serial.print(gauss);
Serial.print(" Gauss ");
Serial.println();

if( gauss < 100){
streetServo.write(0);
} else {
collision();
}
}


void collision() {
streetServo.write(91); //stop moving the road
start = false; //game over state
}

void movePlayer() {
val = analogRead(3);            // reads the value of the potentiometer (value between 0 and 1023)
val = map(val, 0, 1023, 0, 179);  // scale it to use it with the servo (value between 0 and 180)
if((val - oldVal) < -5){  /*check if new value is different from previous with a little threshold
isRight = false;          to avoid swiping the car */
moveServo(50);
}else if(val - oldVal > 5){
isRight = true;
moveServo(65);
}
oldVal = val;
}
```

```
void moveServo(int n) {
  playerServo.write(n);
}

void checkStart() { //this function check if we can start the game or if we are in game over status
  buttonState = digitalRead(buttonPin);
  if(buttonState != lastButtonState ){
    if(buttonState == HIGH){
      start = true;
      if(isRight){
        moveServo(50);
      }else{
        moveServo(65);
      }
      Serial.println("bottone");
    }
  }
  lastButtonState = buttonState;
}

void loop() {
  checkStart();
  if(start) {
    shroom++;
    if(shroom<15){          /*after every collision we give the player a small time where he can
      Serial.println("immunity time");              freely collide with other cars */
      streetServo.write(0);
    }else{
      DoMeasurement();
    }
    movePlayer();
  } else {
    playerServo.write(55);
    shroom = 0;
  }
  debugPotentiometer();
  delay(100);
}


void debugPotentiometer() {
  val = analogRead(3);             // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 179);  // scale it to use it with the servo (value between 0 and 180)
  Serial.print("val ");
  Serial.println(val);
}
```