Dancing Minions

Team: Robot Makers

MeridianTechnoLab

Table of Contents

Project Overview 3
Roles of team members 3
Platform 3
Evolution of the Minion robot 4
First version 4
Second version 5
Third version 6
Minion robot 8
Robot design 8
Construction 8
Step 1. Preparing the platform 8
Step 2. Connecting the power 13
Step 3. Connecting motors 14
Step 4. Connecting the body 15
Step 5. Connecting servo motors (hands) 16
Step 6. Decorating 18
Hardware and Software 22
DC motors control 24
Servo motor control 27
Ultrasonic sensor 29
433MHz radio transmitter and receiver 31
What parts of the project were the hardest? 36
The innovative parts of the robot 36
Credits 36
What's next 36
Conclusion 37
Appendix 1. Source code of the Minion 38

Project Overview

We needed to design a performance including a robot, which is appealing and realistic to make. We decided to make a performance, based on despicable me, as we really like the funny minions. For music we chose happy because it is an upbeat song, which suits the style of our performance. We decided to use an EV3 platform, however we found out that it was too slow and couldn't keep up with our fast-paced song. Therefore, we needed to find a platform which can support motors that are fast enough to keep up with our song.

We decided to use Arduino because we already knew how to use and program it. We already conducted projects with DC motors, the ultrasonic sensor and we were going to experiment with bluetooth/radio communication. All of which we will need in our project, as well as the Arduino software.

Then we had to create several steps of the project:

- Build the base
- Solder the wires together
- Connect the motors, power and Arduino
- The framework to hold up the minion
- Decorating the minion
- Programming

We wanted the robot to be the right size/easy to see (Not so small that it's hard to see), to have an entertaining/eye catching performance as well as to have a good looking robot.

To create our project we needed to learn how to solder, use a drill, program servo motors as well as the motor controller. Resources we needed for project:

- Strong and long-lasting batteries,
- Materials for the minion (fabric, metal for the framework, cotton buds and a polyester ball)
- Motor controller

Roles of team members

All of us (Michael, Daniel, Billy, Connor) took the roles of designers, builders and programmers to create the project. All of us worked on all aspects of the project. However, if one of us was away, the rest of the team would work on the project. Then when we met again we would collaborate and discuss what was done previously. We started using GIT to manage project and collaborate on the source code.

Platform

We decided to use Arduino because we already knew how to use and program it. It allows control of fast DC motors and servo motors. We need DC motors, a motor controller, Li-ion batteries, an ultrasonic sensor and a radio module for robot to robot communication.

Evolution of the Minion robot

First version

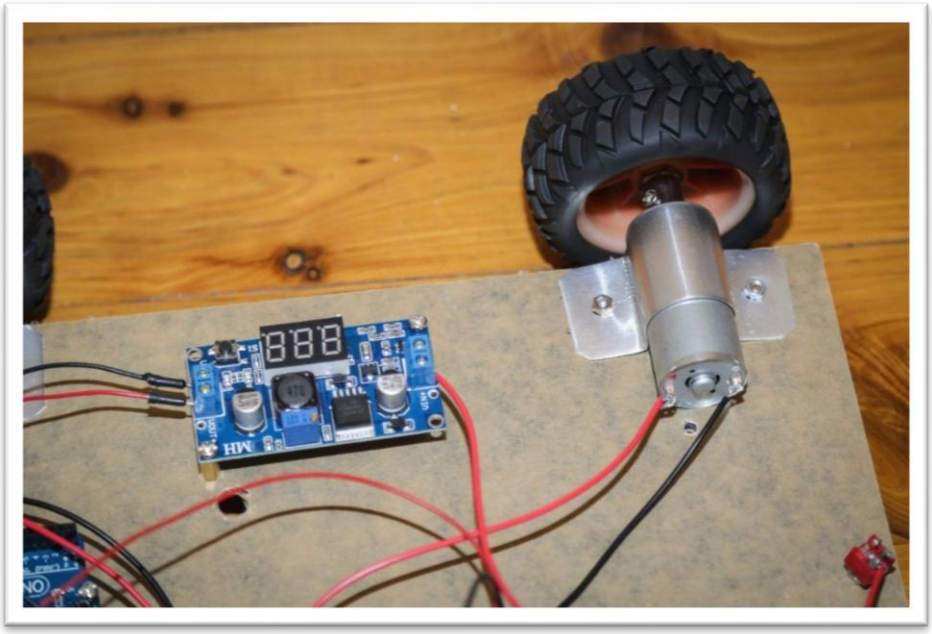
For the first version, we decided to use a 4 wheel drive platform controlled by an Arduino and we made it move to the music happy, also we designed some complex moves for the robot minion. The robot's test drive was successful. Then we decided to make the robots body from cotton buds which was supported by aluminium frame. However, the robot was too tall and too heavy, so it was constantly falling down when the robot was braking or turning.

Second version

Then we decided to use a bigger and heavier aluminium base. We drilled holes, installed motors, controllers, batteries and prepared the robot for the second test. The robot moved too slow because the motors couldn't handle the weight of the robot.

Third version

For the third version, we used more powerful motors, doubled the power supply and we used an acrylic base. When we started the robot, it was fast but after each turn it was getting slower and slower. We noticed that the motor controller was overheating and couldn't handle high current for powerful motors.



Fourth version

Fourth version of the Minions robots performed at the 2019 Australian National Robocup Junior.

We used radio transmitter and receiver for robot to robot communication.

At the competition we had a few issues:

- Hands were falling off
- Mechanical part of power switches were failing and robots could be powered off any time.
- The distance of the radio transmission wasn't long enough and that affected the communication
- We didn't have opportunity to calibrate robots during the test run which could affect the performance

So we planned the following changes and items to resolve issues and make robots safer and more reliable:

- Use gyro sensors to control turns
- Use light sensors to prevent robots from leaving the stage
- Create a function to constantly check distance using ultrasonic sensor to prevent collisions.
- Create a library and move all our functions to it.
- Install reliable switches
- Install LEDs to indicate when initialisation function is completed and robot is ready for the performance.
- Create a light show using LED stripes controlled by Arduino.

Minion robot

Robot design

The arrangement of robot parts: Minion of top of the base so that it can be seen, the controller and electronics on the bottom of the base so that they aren't seen and don't interfere with any moving parts, we put the batteries in the middle so that they move the robot’s centre of gravity there. The DC motors are located near each of the four corners of the base. These motors control the wheel speed and direction.

We also have two servo motors on opposite sides which are used to control the hands. These can turn 180 degrees. We use the ultrasonic sensor which is located at the front of the robot's base to activate the robot and detect any object or person and stop the robot from colliding with it. We use gyro sensor to control turns and the light sensor to avoid robots leaving the stage. All components are well secured, the wires are soldered, the minion has a frame screwed to the base and the rest of the parts are also screwed into place on the base.

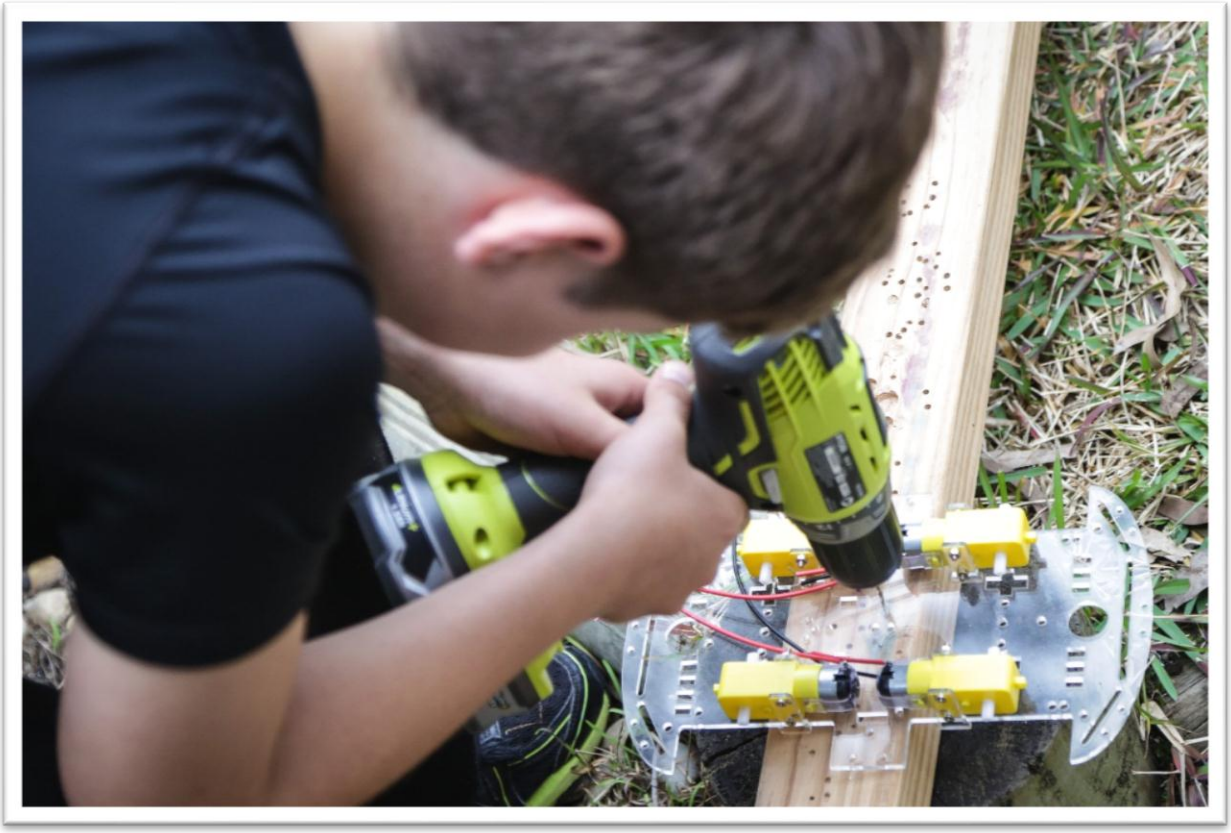
Construction

Step 1. Preparing the platform

The cover was removed from the acrylic base and the wheels were slotted onto the motors.

We soldered wires to the motors.

After marking out where we wanted to place the Arduino and batteries, we drilled holes in the platform so that later we can screw them together.



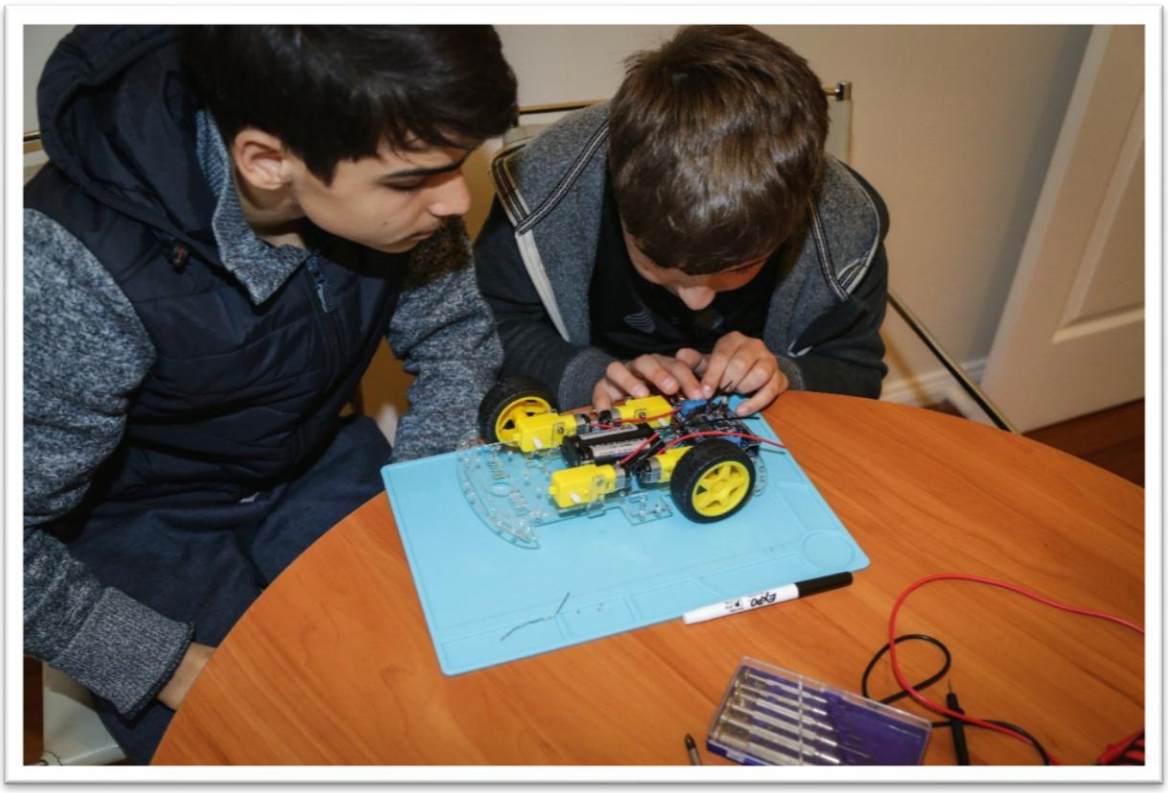
We screwed the battery holder in the middle of the platform.

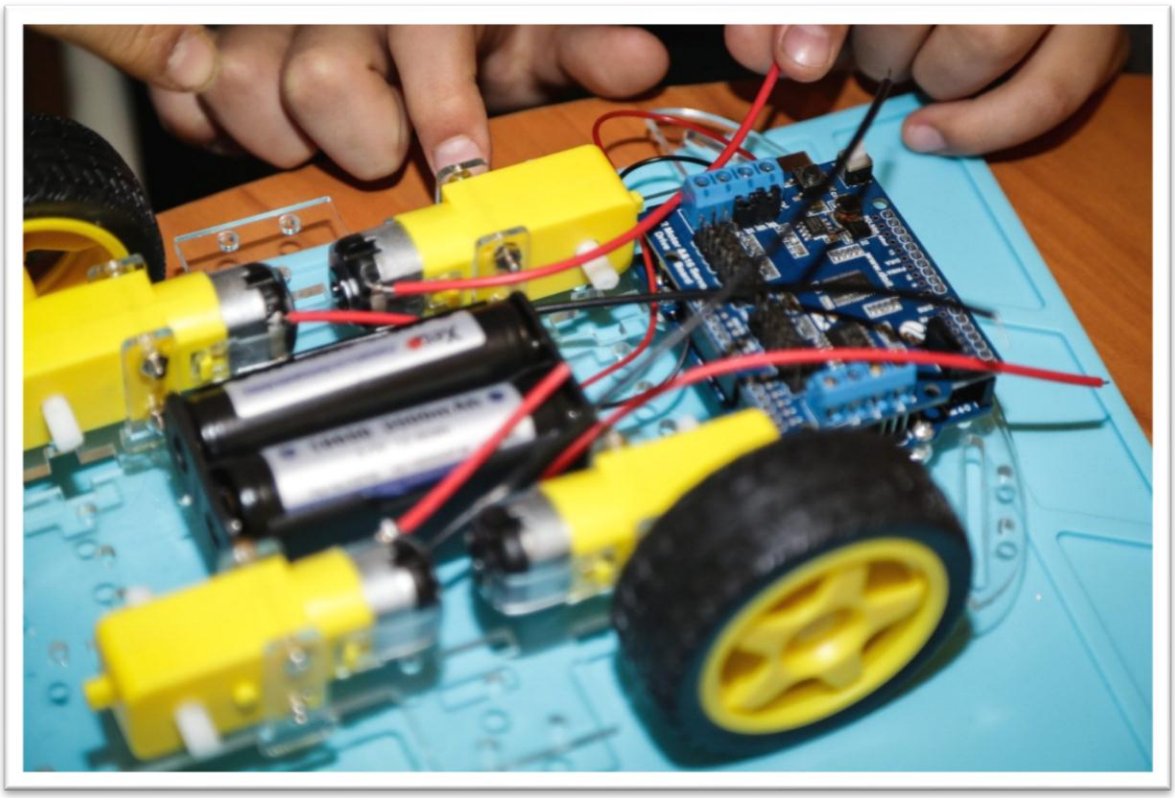
We also placed the Arduino microcontroller in the front of the platform.

All components are installed and we are ready to connect wires and start testing the robot.

Step 2. Connecting the power

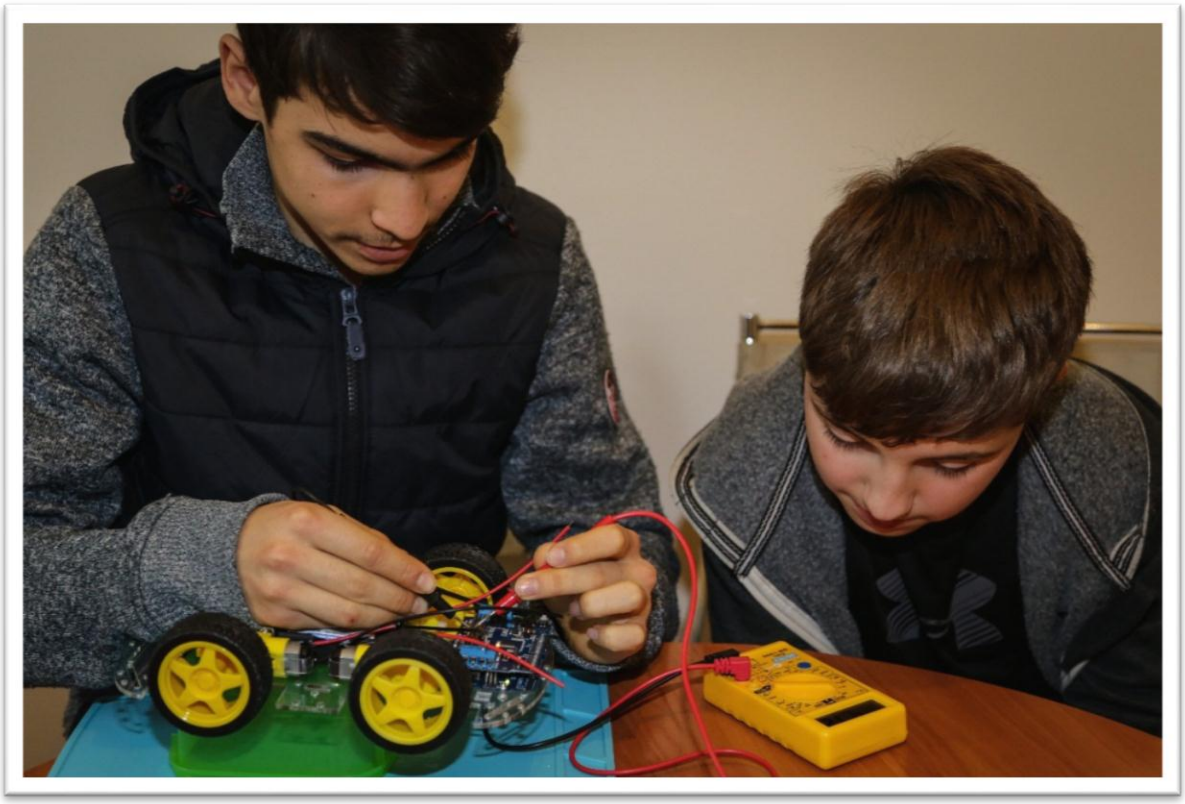
We connected wires from the batteries to the motor controller (black to ground and red to power).





Step 3. Connecting motors

Using a multimeter we made sure that the circuit was working correctly and the batteries were providing power to the motor controller.



First, we connected the front motors to make sure they were both spinning in the right direction (which one wasn't and therefore we had to switch the positions of its wires). Then we connected the back motors to the front ones on the same side, so that they work in unison.

Step 4. Connecting the body

We decided to use a piggy bank made from tin as a body. It’s the perfect size and is light. For the top of the head we will be using a styrofoam sphere which we cut in half.

Step 5. Connecting servo motors (hands)

We cut holes in the minion's body and glued 2 servo motors in them, to control the hands.

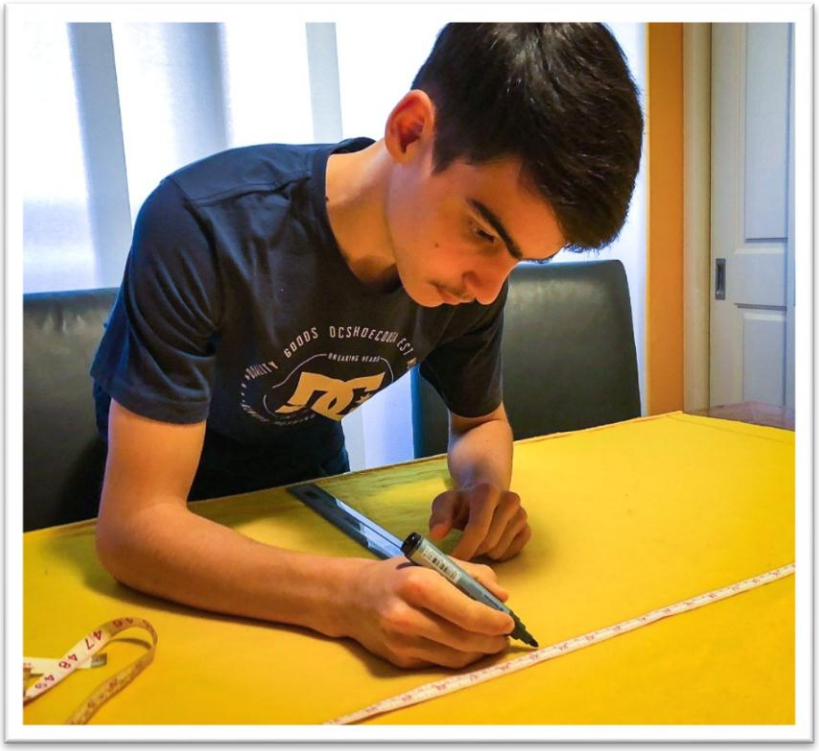
We connected wires from the servo motors to ports 0 and 1 on the motor controller.

We used chopsticks to support the hands. Then we secured servo connectors to them using string, so that the servo motors can move them.

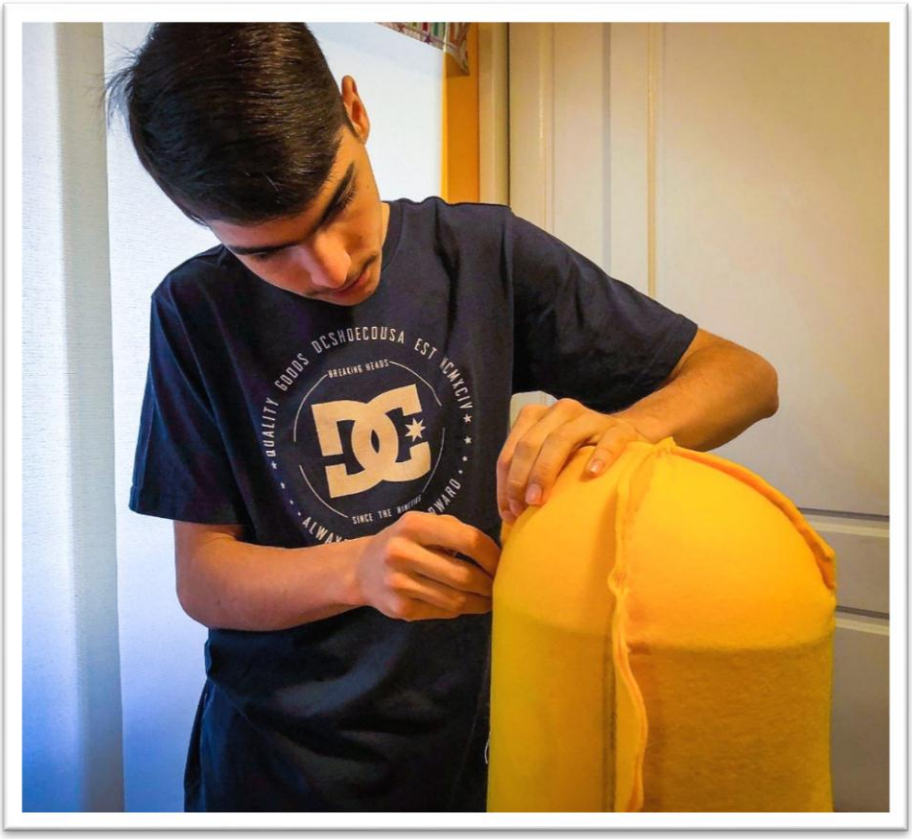
Step 6. Decorating

The circumference of the robot was measured so that we knew where to cut the fabric.

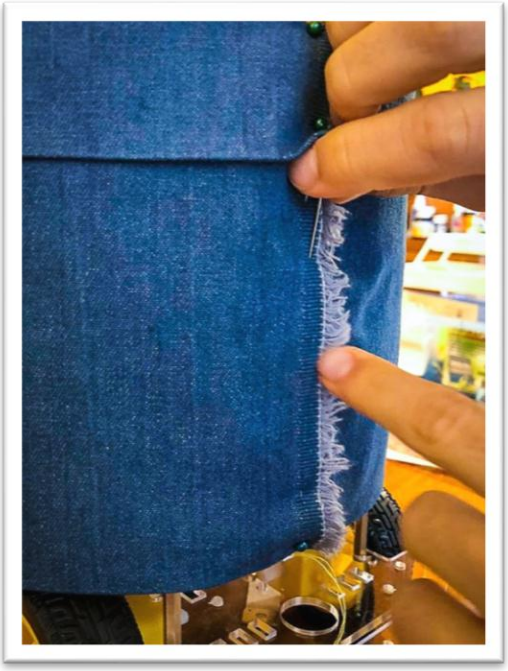
Lines were marked on the fabric according to the measurements i.e. Since the circumference was 30cm, a distance 30cm from the side of the fabric was marked.



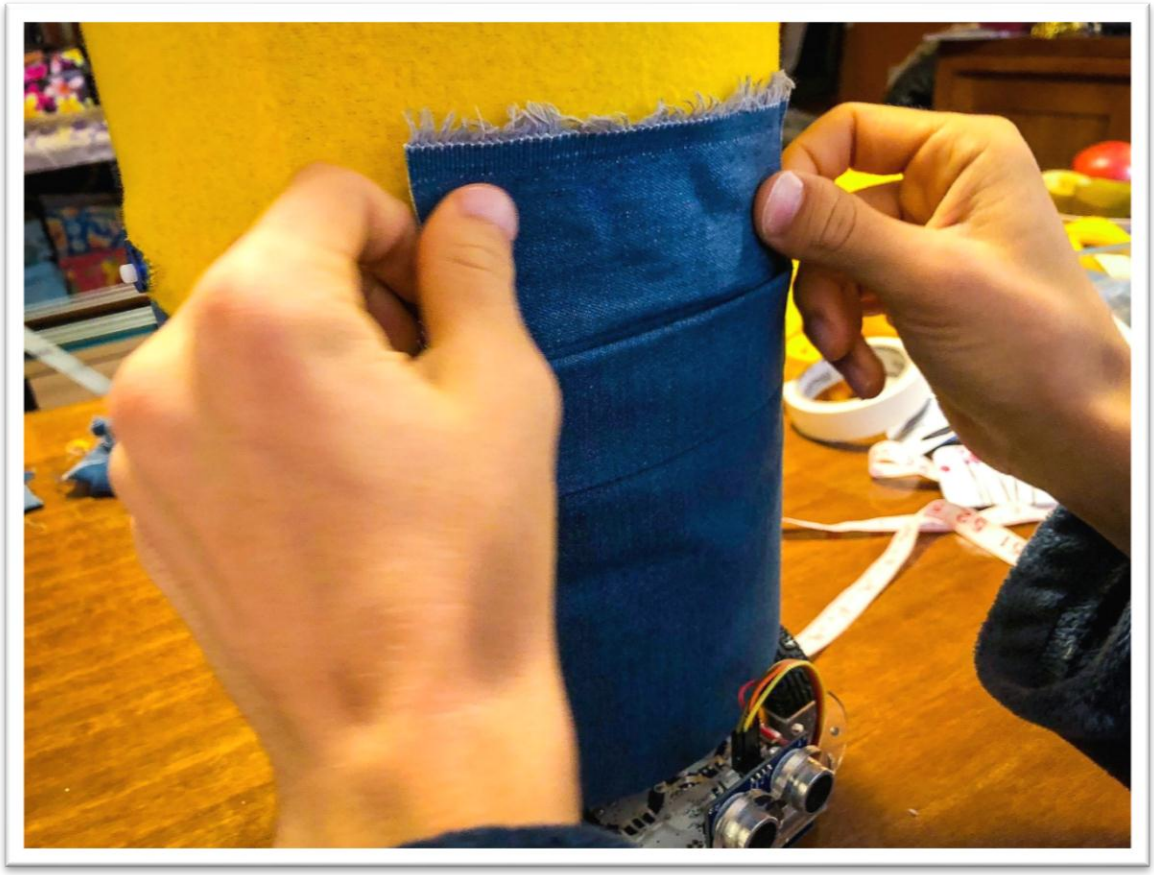
After the fabric was put on the minion inside out. The overlapping parts were hand-sewn together.



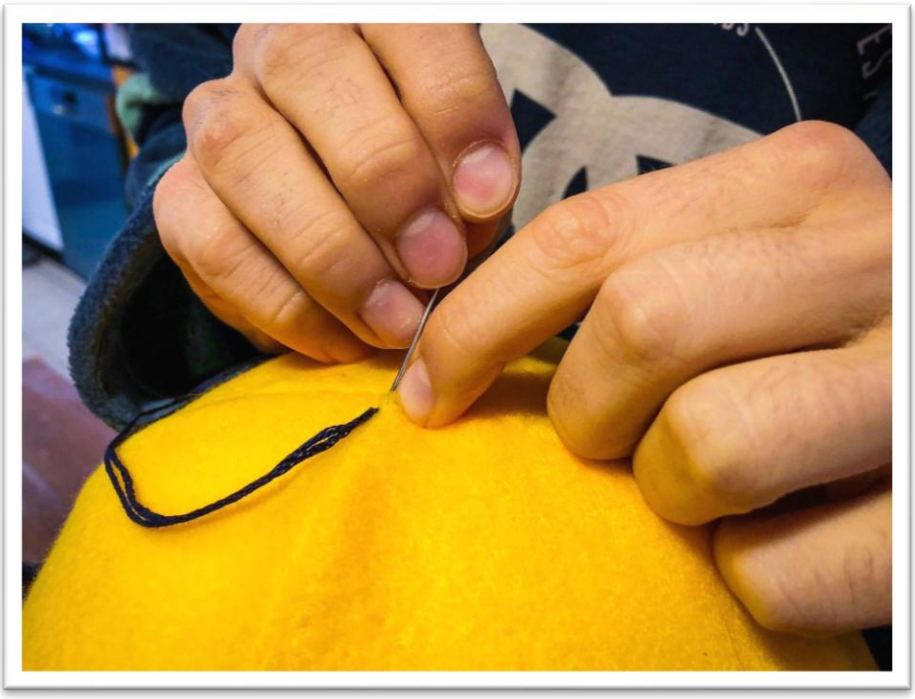
he same was done to the trousers (marking and cutting) however, instead of sewing the fabric together. We ironed the folds so that they stayed together and used pins to keep them around the minion.



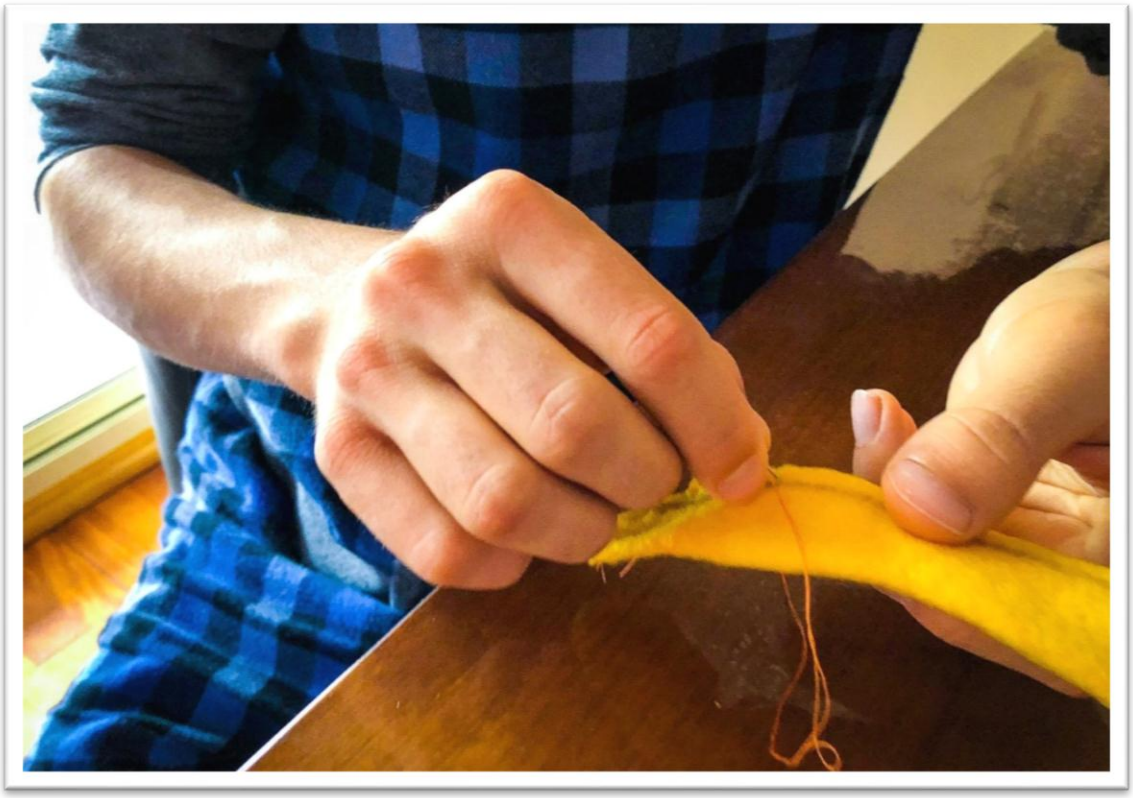
A piece of fabric was tucked into the minion’s pants.



String was sewed onto the minion’s head as hair.



After marking and cutting the fabric for the hands, the overlap was sewn together.



The arm was turned inside out and stuffed with polyester.

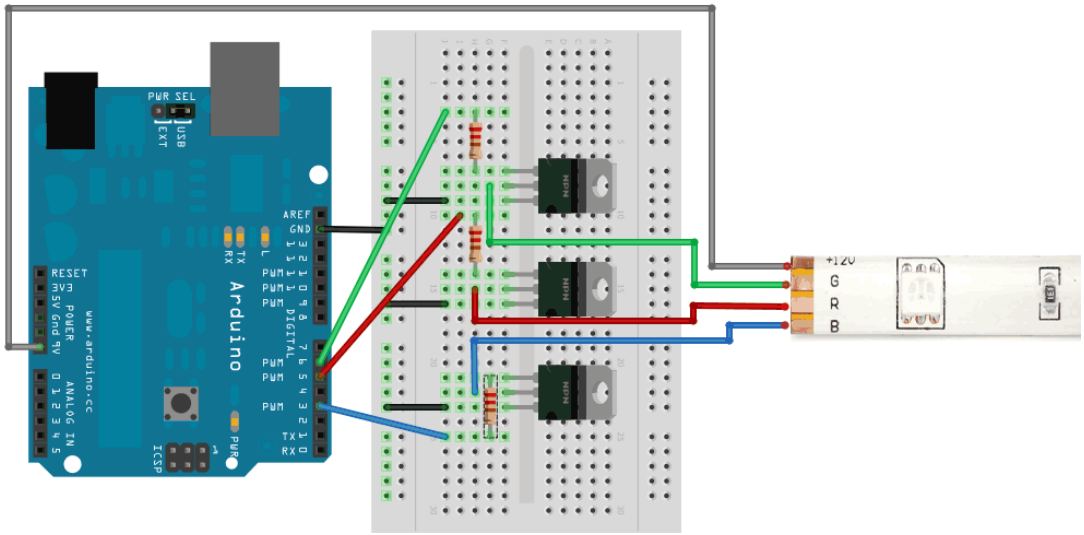
Interactive Light Show

To make our performance more interesting we decided to create the interactive Light Show board which will be flashing with the music and changing colours.

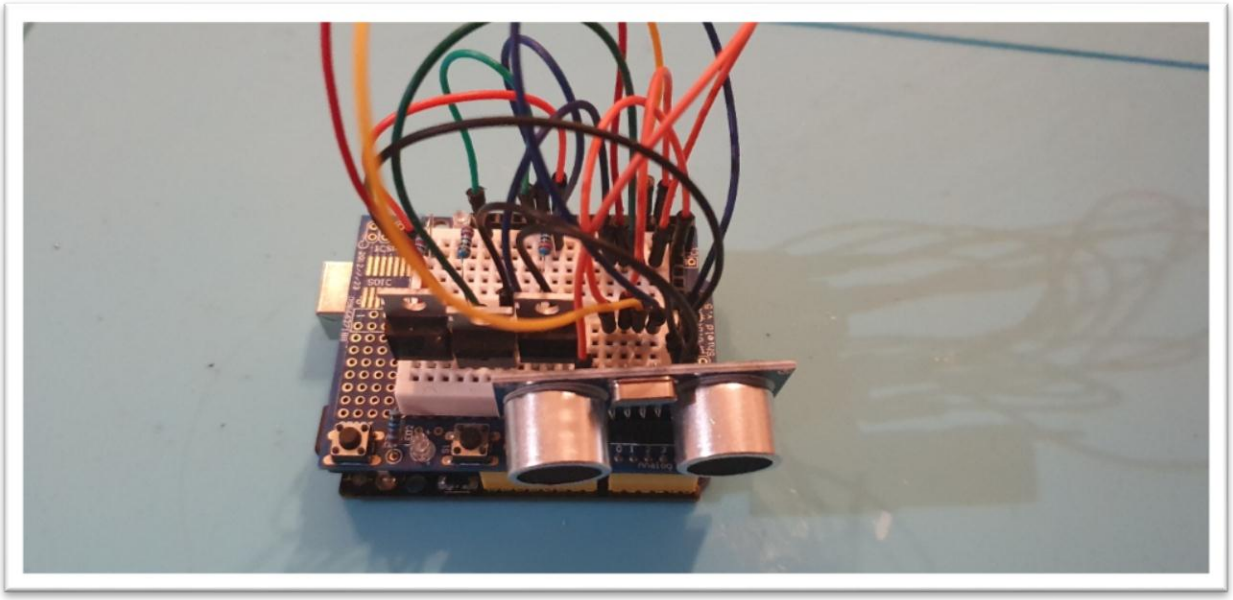
We created the frame from timber and secured LED strips to the back wall.

The 5m LED stripe has 500 LEDs and need about 1 Ampere per channel – red, green, blue.

Therefore, we had to use powerful MOSFET transistors to control that current and supply it to the LEDs:



We prototyped and tested the circuit:



We use the ultrasonic sensor to activate the Light Show.

As part of the preparation we will place the board in the middle of the stage and will cover it with our posters. When the music starts we will remove the poster which will activate the Light Show and it will start flashing to the music.

The source code is included in Appendix 3.

Hardware and Software

We decided to use Arduino because we already knew how to use and program it. We have already completed projects with motors and ultrasonic sensors and we wanted to learn more about servo motors and radio communication.

Components we used to build the minion robot:

N	Name	Qty
1	Arduino UNO	1
2	2 Motor and 16 Servo Drive Board	1
3	HC-SR04 Ultrasonic Sensor	1
4	Tower Pro SG90 Servo Motor	2
5	Gyro sensor MPU6050	1
6	Light sensor	1
7	LED	1
8	Resistor 220 Ohm	1
9	Switch	1
10	TT DC Motor	4
11	Wheel	4
12	433 MHz RF Transmitter and Receiver	1
13	Li-ion 18650 Battery	2
14	Li-ion 18650 Battery Holder	1
15	Tin Can (Body)	1
16	styrofoam Ball (Top of the head)	1

DC motors control

We used a DC motor controller which has 2 ports which can support two DC motors. There are four pins which are used to control them, these are DIR A which controls the direction of one motor and PWM A which controls the speed of the same motor. The other two pins do the same for the second motor. We created functions to control these motors so that we only have to type in the speed and direction. This also gives us flexibility if we decide to change the function, as we will only have to change one piece of code instead of lines and lines of it.

```
// Move robot forward
void RobotMoveForward(int iSpeed, int iTime) {
  digitalWrite(MotorDirA, HIGH);
  analogWrite(MotorPWMA, iSpeed);
  digitalWrite(MotorDirB, HIGH);
  analogWrite(MotorPWMB, iSpeed);
  delay(iTime);
}
```

```
// Move robot back
void RobotMoveBack(int iSpeed, int iTime) {
  digitalWrite(MotorDirA, LOW);
  analogWrite(MotorPWMA, iSpeed);
  digitalWrite(MotorDirB, LOW);
  analogWrite(MotorPWMB, iSpeed);
  delay(iTime);
}

// Turn robot Left
void RobotTurnLeft(int iSpeed, int iTime) {
  digitalWrite(MotorDirA, HIGH);
  analogWrite(MotorPWMA, iSpeed);
  digitalWrite(MotorDirB, LOW);
  analogWrite(MotorPWMB, iSpeed);
  delay(iTime);
}

// Turn robot left degrees
void RobotTurnLeftDegrees(int iDegrees) {
  digitalWrite(MotorDirA, HIGH);
  analogWrite(MotorPWMA, 200);
  digitalWrite(MotorDirB, LOW);
  analogWrite(MotorPWMB, 200);
  delay(K_Left * iDegrees);
}

// Turn robot right
void RobotTurnRight(int iSpeed, int iTime) {
  digitalWrite(MotorDirA, LOW);
  analogWrite(MotorPWMA, iSpeed);
```

```
  digitalWrite(MotorDirB, HIGH);
  analogWrite(MotorPWMB, iSpeed);
  delay(iTime);
}

// Turn robot right degrees
void RobotTurnRightDegrees(int iDegrees) {
  digitalWrite(MotorDirA, LOW);
  analogWrite(MotorPWMA, 200);
  digitalWrite(MotorDirB, HIGH);
  analogWrite(MotorPWMB, 200);
  delay(K_Right * iDegrees);
}

// Smooth turn robot
void RobotSmoothTurn(int iSpeedA, int iSpeedB, int iTime) {
  digitalWrite(MotorDirA, HIGH);
  analogWrite(MotorPWMA, iSpeedA);
  digitalWrite(MotorDirB, HIGH);
  analogWrite(MotorPWMB, iSpeedB);
  delay(iTime);
}

// Stop robot
void RobotStop(int iTime) {
  analogWrite(MotorPWMA, 0);
  analogWrite(MotorPWMB, 0);
  delay(iTime);
}
```

Servo motor control

We used 2 TowerPro SG90 servo motors to control robot's hands. Motor controller allows control up to 16 servo motors. We only used ports 0 and 1.

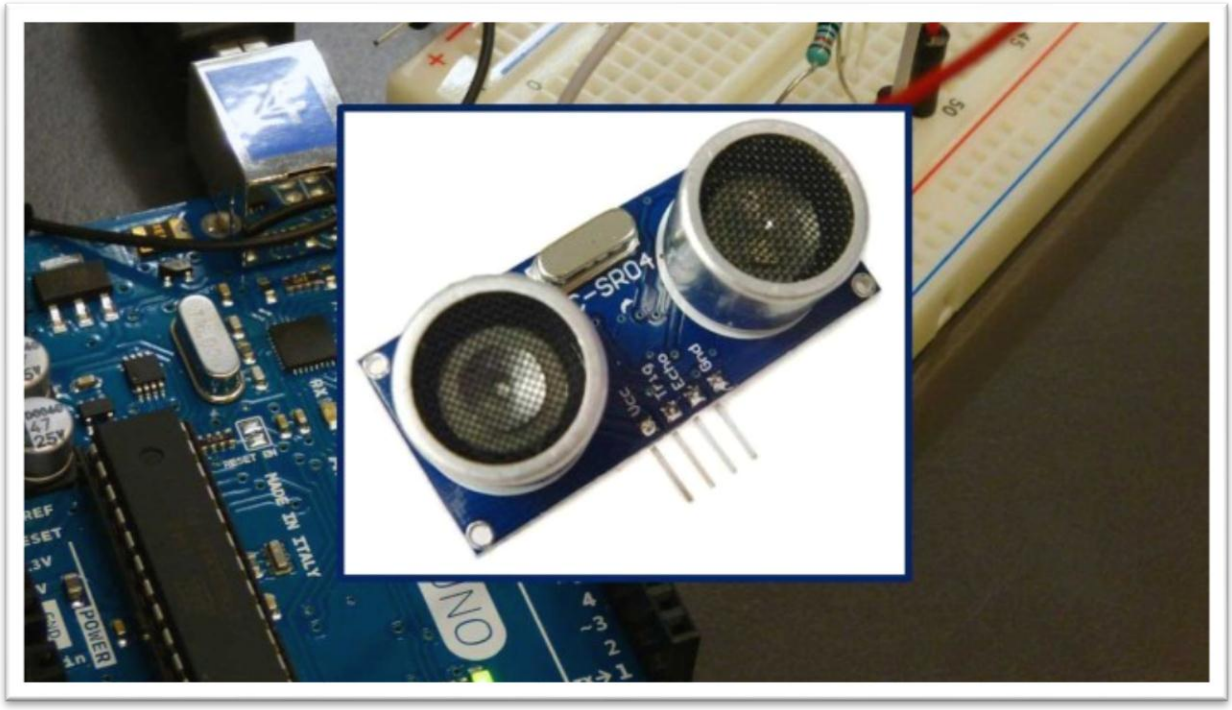
We only used three positions for the hands – up, forward and down. Therefore, we decided to create functions to move the hands to these positions:

```
// Left hand up
void LeftHandUp() {
  pwm.setPWM(1, 0, 150);
}

// Left hand down
void LeftHandDown() {
  pwm.setPWM(1, 0, 600);
}

// Left hand forward
void LeftHandForward() {
  pwm.setPWM(1, 0, 320);
}

// Right hand up
void RightHandUp() {
  pwm.setPWM(0, 0, 600);
}

// Right hand down
void RightHandDown() {
  pwm.setPWM(0, 0, 150);
}
```

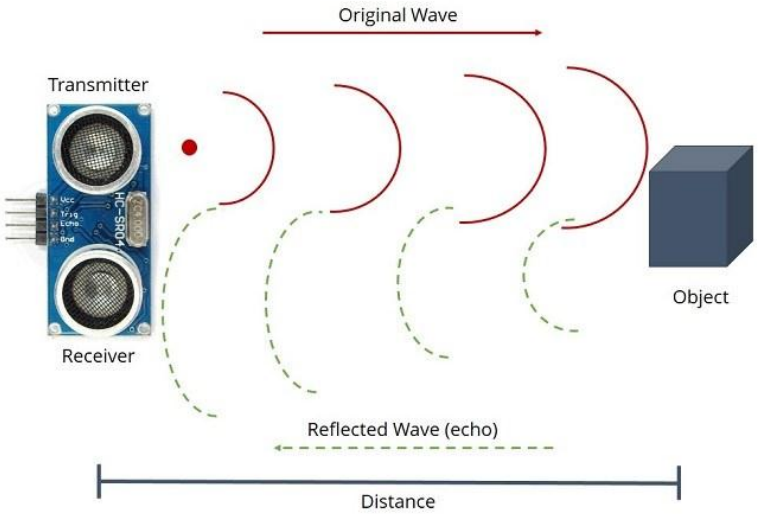
```
// Right hand forward
void RightHandForward() {
  pwm.setPWM(0, 0, 350);
}

// Both hands up
void HandsUp() {
  LeftHandUp();
  RightHandUp();
}

// Both hands down
void HandsDown() {
  LeftHandDown();
  RightHandDown();
}

// Both hands forward
void HandsForward() {
  LeftHandForward();
  RightHandForward();
}
```

Ultrasonic sensor

We are using the ultrasonic sensor to start our program when it sees our hand approximately 5-40 cm away from it. We also want to use it to stop any collision by not allowing the robot to move forward when it sees an object in front of it, which will protect the robot and humans.



The ultrasonic sensor calculates distance by sending a signal with its transmitter (trigger pin) in the form of a high-frequency sound. When this hits an object, it's reflected and comes back to be received by the receiver (echo pin).



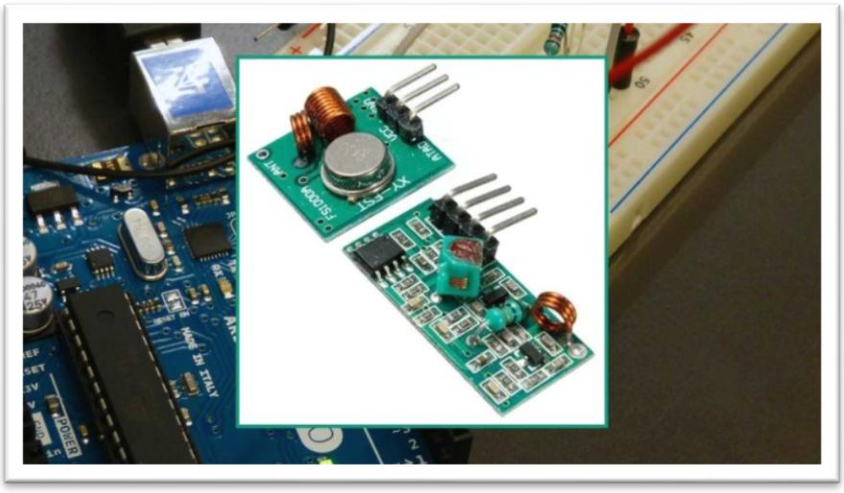
Then, the time between sending and receiving the signal is used to calculate the distance between the sensor/object. Since we know the speed of sound, it means that the processor can use the following formula to calculate the distance. Distance = Speed x Time.

We created the function “UltraSonicActivation” for the first robot to wait until it sees our hands:

```
// Ultrasonic activation
void UltraSonicActivation() {
  int uS = sonar.ping_cm();
  delay(50);
  while ((uS == 0) || (uS > 30)) {
    uS = sonar.ping_cm();
    Serial.print("Activation: ");
    Serial.print(uS);
    Serial.println("cm");
    delay(50);
  }
  Serial.println("US: Start dancing");
}
```

433MHz radio transmitter and receiver

We decided to use a 433MHz radio transmitter and receiver for robot to robot communication. When the first robot is activated using the ultrasonic sensor it will send another robot the “start dancing” command using its 433MHz transmitter. We also noticed that one robot is faster than the other, so we decided to use radio signals to synchronise both robots.



We created the “SendStart” function in the program for the first minion to send the command to the second robot:

```
// Tell another minion to start dancing
void SendStart() {
  const char *msg = "start";
  radio.send((uint8_t *)msg, strlen(msg));
  radio.waitPacketSent();
  delay(50);
  Serial.println("RF: Sent start");
}
```

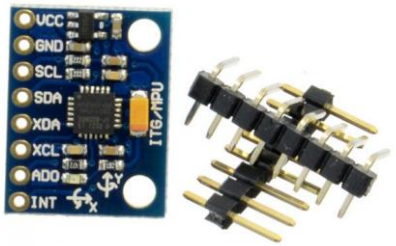
We also created the “RadioActivation” function in the program for the second Minion to wait until the Start command received:

```
// Radio activation
void RadioActivation() {
  uint8_t buf[5];
```

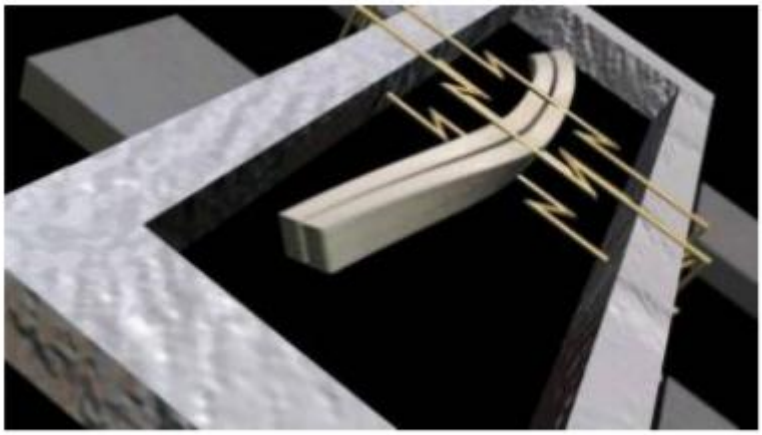
```
  uint8_t buflen = 5;
  String st = "     ";
  while (st.substring(0, 5) != "start") {
    if (radio.recv(buf, &buflen))
    {
      st = (char*)buf;
    }
    delay(50);
  }
  Serial.println("RF: Start dancing");
}
```

Gyro sensor

We use gyro sensor MPU6050 to accurately control the angle to which the robot will turn.



The MPU 6050 is a 6 DOF (Degrees of Freedom) or a six axis IMU sensor, which means that it gives six values as output. Three values from the accelerometer and three from the gyroscope. The MPU 6050 is a sensor based on MEMS (Micro Electro Mechanical Systems) technology. Both the accelerometer and the gyroscope is embedded inside a single chip. This chip uses I2C (Inter Integrated Circuit) protocol for communication.



Piezo Electric Gyroscope

Gyroscopes work on the principle of Coriolis acceleration. Imagine that there is a fork like structure, that is in constant back and forth motion. It is held in place using piezo electric crystals. Whenever, you try to tilt this arrangement, the crystals experience a force in the direction of inclination.

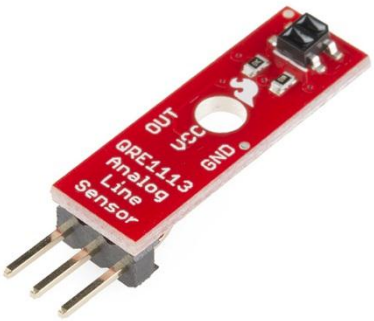
This is caused as a result of the inertia of the moving fork. The crystals thus produce a current in consensus with the piezo electric effect, and this current is amplified.

Below is the part of the source code which reads values from the gyroscope and allows to turn right to a specific angle:

```
//Turn Robot to a gyro angle
void Minion::GyroTurnRightDegrees(int iDegrees) {
  iDegrees = iDegrees * G_Right;
  float StartAngle = 0;
  float Angle = 0;
  float diff = 0;

  mpu6050.update();
  StartAngle = mpu6050.getAngleZ();
  Angle = StartAngle;

  digitalWrite(MotorDirA, LOW);
  analogWrite(MotorPWMA, 200);
  digitalWrite(MotorDirB, HIGH);
  analogWrite(MotorPWMB, 200);

  while ((Angle - StartAngle) < iDegrees) {
    mpu6050.update();
    Angle = mpu6050.getAngleZ();
    diff = Angle - StartAngle;
  }
  RobotStop(100);

}
```

Light sensor

We used the light sensor QRE1113 to avoid the robot leaving the stage.



The board's QRE1113 IR reflectance sensor is comprised of two parts - an IR emitting LED and an IR sensitive phototransistor. When you apply power to the VCC and GND pins the IR LED inside the sensor will illuminate. A 100O resistor is on-board and placed in series with the LED to limit current. A 10kO resistor pulls the output pin high, but when the light from the LED is reflected back onto the phototransistor the output will begin to go lower. The more IR light sensed by the phototransistor, the lower the output voltage of the breakout board.

Below is the part of the source code which reads values from the light sensor:

```
int Colour = analogRead(LightSensorPin);
  if (Colour > 990){
    RobotMoveBackCM(45);
  }
```

What parts of the project were the hardest?

The hardest part of the project was making sure the robot could still move and dance. When we started the program after putting the body on the first version, for the first time, it simply kept falling backwards.

Hence, we decided to make a lighter body, make the robot's centre of gravity as close the middle and low as possible (by placing the heaviest object such as batteries there).

We also had to change the robot's routine as it couldn't cope with the turns. If we could start again, we would definitely make the robot first and then design the routine as we had to throw the first version out because of physical limitations.

In the future we plan to use a gyro sensor to calculate how far the robot has turned, as well as to be able to control the arms (Putting a gyro sensor along with a transmitter into our gloves, allowing the robot to mirror our arm movements using a receiver).

The innovative parts of the project

- We use gyro sensor to accurately turn left or right to a specific angle.
- Instead of using standard delay() function we created the pause function which allows robot to check sensors (light sensor and ultrasonic sensor) and avoid collisions and prevent the robot from leaving the stage area.
- Interactive Light Show board which can be activated using ultrasonic sensor and flashing lights with music changing colours.
- We created the library and used object oriented language to call functions
- We made robots from everyday materials such as tin, acrylic, fabric, chopsticks etc.
- We used feedback loops to allow robots to process data from sensors constantly.
- We made the robots from separate electronic components, not commercial kits.

Credits

All of us (Michael, Daniel, Billy, Connor) designed, built and programmed to create the robot. All of us worked on each step of the project. However, if one of us was away, the rest of the team would work on the project. Then when we met again we would collaborate and discuss what was done previously. We were using GIT and GITHUB for version control and to collaborate on the source code.

What's next

We would like to make a few changes in the future:

1. Improve "parallel processing" to read values from sensors and process using interruptions.
2. Use gyro sensors in our gloves to manually control robot's hands.

Conclusion

It took about 15 months to experiment and build two minions and program them.

During this time we have learnt a lot:

- how to design robots and manage projects;
- how to solder, drill and cut acrylic and aluminium sheets;
- how to calibrate and control DC and servo motors
- how to use radio transmitter and receiver
- problem solving and debugging
- how to use GIT and GITHUB to collaborate on the source code.

It was lots of fun although at times frustrating, but we are very happy with our minion robots!

Appendix 1. Source code of the Minion

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
//
//  Minion1.ino - Dancing Minion 1
//
//  ver. 5.1
//  Last updated: 16/06/2019
//
//
//  14/06/2019 - Added comments
//
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////


#include <Minion.h>

//          LHD, RHD, LHU, RHU, LHF, RHF, G_R ,  G_L , K_F     , K_B
Minion minion1(460, 105, 120, 470, 270, 275, 1.70, -1.70, 1770/100, 1785/100);

void Hands() {
  minion1.HandsUp();
  minion1.pause(400);
  minion1.RightHandForward();
  minion1.pause(400);

  for (int i=1; i <= 4; i++){
    minion1.RightHandUp();
    minion1.LeftHandForward();
    minion1.pause(400);
    minion1.RightHandForward();
```

```
  minion1.LeftHandUp();
  minion1.pause(400);
}

minion1.HandsDown();
minion1.pause(800);
//RadioActivation();
minion1.RightHandUp();
minion1.pause(800);

for (int i=1; i <= 4; i++){
  minion1.RightHandDown();
  minion1.LeftHandUp();
  minion1.pause(800);
  minion1.RightHandUp();
  minion1.LeftHandDown();
  minion1.pause(800);
}

minion1.HandsDown();
minion1.pause(800);

for (int j = 1; j <= 3; j++) {
  minion1.GyroTurnRightDegrees(100);
  minion1.RobotStop(500);
  minion1.RightHandUp();
  minion1.pause(400);
  for (int i=1; i <= 2; i++){
    minion1.RightHandDown();
    minion1.LeftHandUp();
    minion1.pause(400);
    minion1.RightHandUp();
    minion1.LeftHandDown();
```

```
      minion1.pause(400);
    }
    minion1.HandsDown();
    minion1.pause(800);
  }
  minion1.GyroTurnRightDegrees(100);
  minion1.RobotStop(500);
}

void Happy() {
  minion1.start();
  delay(600);

  // Begining
  minion1.RobotMoveForwardCM(50);
  minion1.RobotMoveForwardCM(50);
  minion1.RobotStop(500);
  minion1.HandsUp();
  minion1.pause(500);
  minion1.HandsDown();
  minion1.pause(500);
  minion1.GyroTurnLeftDegrees(90);
  minion1.RobotStop(500);
  minion1.HandsUp();
  minion1.pause(500);
  minion1.HandsDown();
  minion1.pause(500);
  minion1.RobotMoveForwardCM(30);
  minion1.RobotStop(500);
  minion1.RightHandUp();
  minion1.pause(500);
  minion1.LeftHandUp();
  minion1.RightHandDown();
```

```
minion1.pause(500);
minion1.LeftHandDown();
minion1.RightHandUp();
minion1.pause(500);
minion1.LeftHandUp();
minion1.RightHandDown();
minion1.pause(500);
minion1.LeftHandDown();
minion1.pause(500);
minion1.GyroTurnRightDegrees(90);
minion1.RobotStop(500);
minion1.RobotMoveBackCM(50);
minion1.RobotStop(500);
minion1.GyroTurnRightDegrees(90);
minion1.RobotStop(100);
//BluetoothActivation();
delay(1200);
minion1.HandsUp();
minion1.pause(250);
minion1.GyroTurnRightDegrees(90);
minion1.RobotStop(500);
minion1.HandsDown();
//BluetoothActivation();
delay(1000);
minion1.RobotMoveForwardCM(50);
delay(300);
minion1.RobotStop(100);
delay(500);
minion1.HandsUp();
delay(2000);
minion1.HandsDown();
delay(500);
minion1.RobotMoveBackCM(50);
```

```
delay(300);
minion1.RobotStop(100);
//BluetoothActivation();
delay(1000);

// Turns 7s
minion1.HandsUp();
delay(500);
minion1.HandsDown();
minion1.pause(500);
minion1.GyroTurnRightDegrees(90);
minion1.RobotStop(1000);
minion1.HandsUp();
minion1.pause(500);
minion1.HandsDown();
minion1.pause(500);
minion1.GyroTurnRightDegrees(90);
minion1.RobotStop(500);

// Hands 23s
Hands();
//Bluetooth Activation();
delay(1000);

// Final moves
minion1.RobotMoveForwardCM(40);
minion1.RobotStop(500);
minion1.HandsUp();
minion1.pause(500);
minion1.HandsDown();
minion1.pause(1000);
minion1.RobotMoveBackCM(50);
minion1.RobotStop(500);
```

```
delay(1000);
minion1.HandsUp();
delay(700);
minion1.HandsDown();
minion1.pause(1000);
minion1.RobotMoveForwardCM(50);
minion1.RobotStop(500);
minion1.HandsUp();
minion1.pause(500);
minion1.HandsDown();
minion1.pause(1000);
delay(000);

// 8 sec
minion1.GyroTurnLeftDegrees(100);
minion1.RobotStop(500);
minion1.RightHandForward();
minion1.pause(500);
minion1.LeftHandForward();
minion1.RightHandDown();
minion1.pause(500);
minion1.LeftHandDown();
minion1.RightHandForward();
minion1.pause(500);
minion1.LeftHandUp();
minion1.pause(500);
minion1.RightHandUp();
minion1.LeftHandForward();
minion1.pause(500);
minion1.RightHandForward();
minion1.LeftHandUp();
minion1.pause(500);
minion1.HandsDown();
```

```
  minion1.pause(500);
  minion1.GyroTurnRightDegrees(100);
  minion1.RobotStop(500);
  minion1.HandsForward();
  minion1.pause(200);
  delay(50);
  minion1.HandsUp();
  delay(150);
  minion1.HandsDown();
  delay(50);
  minion1.HandsUp();
  delay(150);
  minion1.HandsDown();
  delay(50);
  minion1.HandsUp();
  delay(100);
  minion1.HandsDown();
  delay(50);
  minion1.HandsUp();
  delay(100);
  minion1.HandsDown();
  delay(50);
  minion1.HandsUp();
  delay(50);
  minion1.HandsDown();
  delay(50);
  minion1.HandsUp();
  delay(50);
  minion1.HandsDown();


}
```

```
void setup() {
  Happy();
}

void loop(){
}
```

Appendix 2. Source code of the library

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//
//
//
//  Minion.h - Library for dancing minions
//
//  ver. 1.1
//  Last updated: 16/06/2019
//
//
//  14/06/2019 - Added comments
//
//
////////////////////////////////////////////////////////////////////////////////////////////////////
//



#ifndef minion
#define minion

#if (ARDUINO >= 100)
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

class Minion {

 public:
```

```
  Minion(int _lHandDown, int _rHandDown, int _lHandUp, int _rHandUp, int _lHandForward, int
_rHandForward, float _G_Right, float _G_Left, float K_Forward, float K_Backward);

  void start();

  void UltraSonicActivation();
  void pause(int interval);
  void RobotStop(int iTime);
  void RobotMoveForwardCM(int iCM);
  void RobotMoveBackCM(int iCM);

  //Left Hand Control
  void LeftHandUp();
  void LeftHandDown();
  void LeftHandForward();

  //Right Hand Control
  void RightHandUp();
  void RightHandDown();
  void RightHandForward();

  //All Hand Control
  void HandsUp();
  void HandsDown();
  void HandsForward();

  //Turning
  void GyroTurnRightDegrees(int iDegrees);
  void GyroTurnLeftDegrees(int iDegrees);
```

```
  private:

};

#endif
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//
//
//
//  Minion.cpp - Library for dancing minions
//
//  ver. 1.1
//  Last updated: 16/06/2019
//
//
//  14/06/2019 - Added comments
//
//
////////////////////////////////////////////////////////////////////////////////////////////////////
//



#include "Minion.h"
#include <ServoDriver.h>
#include <NewPing.h>
#include <Wire.h>
#include <MPU6050_tockn.h>

ServoDriver pwm = ServoDriver();

unsigned long previousMillis = 0;

MPU6050 mpu6050(Wire);



int TRIGGER_PIN = 9;
int ECHO_PIN = 8;
```

```
int MAX_DISTANCE = 200;
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

// Motor A
int MotorDirA  = 12;
int MotorPWMA  = 11;

// Motor B
int MotorDirB  = 7;
int MotorPWMB  = 6;

int LED_PIN = 5;

int LightSensorPin = A0;
int LightSensor;

// Tells the Servo Motors how far to turn
int lHandDown;
int rHandDown;
int lHandUp;
int rHandUp;
int lHandForward;
int rHandForward;

// Used to turn and move forward, Passed in through the instatiation of the minion object
float G_Right;
float G_Left;
float K_Forward;
float K_Backward;
```

```
Minion::Minion(int _lHandDown, int _rHandDown, int _lHandUp, int _rHandUp, int _lHandForward,
int _rHandForward, float _G_Right, float _G_Left, float _K_Forward, float _K_Backward){
  lHandDown = _lHandDown;
  rHandDown = _rHandDown;
  lHandUp = _lHandUp;
  rHandUp = _rHandUp;
  lHandForward = _lHandForward;
  rHandForward = _rHandForward;

  G_Right = _G_Right;
  G_Left = _G_Left;
  K_Forward = _K_Forward;
  K_Backward = _K_Backward;

}


void Minion::start(){
  //Making the LED off so we know that the robot is not ready
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);

  // DC Motors
  pinMode(MotorDirA, OUTPUT);
  pinMode(MotorPWMA, OUTPUT);
  pinMode(MotorDirB, OUTPUT);
  pinMode(MotorPWMB, OUTPUT);

  // Ultrasonic Sensor
  Serial.begin(9600);
```

```
  //Servo Motor
  pwm.begin();
  pwm.setPWMFreq(50);
  HandsDown();
  delay(500);

  //Gyro
  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);

  //LED on saying that the robot is ready
  digitalWrite(LED_PIN, HIGH);

  UltraSonicActivation();
}

void Minion::UltraSonicActivation() {
   int uS = sonar.ping_cm();
   delay(50);
   while ((uS == 0) || (uS > 30)) {
      uS = sonar.ping_cm();
      delay(50);
   }
   Serial.println("Start dancing");
}

void Minion::pause(int interval){
    int uS = sonar.ping_cm();
    unsigned long currentMillis = millis();
    int Colour = analogRead(LightSensorPin);
```

```
    while(currentMillis - previousMillis <= interval){
     currentMillis = millis();
     int uS = sonar.ping_cm();
     int Colour = analogRead(LightSensorPin);
     if ((uS > 0) && (uS < 15)){
      RobotStop(20);

      while ((uS > 0) && (uS < 15)){
       uS = sonar.ping_cm();
       delay(400);
      }
     } else {
      if (Colour > 990){
       RobotMoveBackCM(45);
      }
     }
    }
    previousMillis = currentMillis;
}

void Minion::RobotStop(int iTime) {
  analogWrite(MotorPWMA, 0);
  analogWrite(MotorPWMB, 0);
  pause(iTime);
}

void Minion::RobotMoveForwardCM(int iCM) {
  int iTime = iCM * K_Forward;
  digitalWrite(MotorDirA, HIGH);
  analogWrite(MotorPWMA, 200);
```

```
  digitalWrite(MotorDirB, HIGH);
  analogWrite(MotorPWMB, 200);
  pause(iTime);
}
void Minion::RobotMoveBackCM(int iCM) {
  int iTime = iCM * K_Backward;
  digitalWrite(MotorDirA, LOW);
  analogWrite(MotorPWMA, 200);
  digitalWrite(MotorDirB, LOW);
  analogWrite(MotorPWMB, 200);
  pause(iTime);
}

void Minion::LeftHandUp(){
 pwm.setPWM(1, 0, lHandUp);
}

void Minion::LeftHandDown(){
 pwm.setPWM(1, 0, lHandDown);
}

void Minion::LeftHandForward(){
 pwm.setPWM(1, 0, lHandForward);
}

void Minion::RightHandUp() {
 pwm.setPWM(0, 0, rHandUp);
}

void Minion::RightHandDown() {
 pwm.setPWM(0, 0, rHandDown);
```

```
}

void Minion::RightHandForward() {
  pwm.setPWM(0, 0, rHandForward);
}

void Minion::HandsUp() {
  LeftHandUp();
  RightHandUp();
}

void Minion::HandsDown() {
  LeftHandDown();
  RightHandDown();
}

void Minion::HandsForward() {
  LeftHandForward();
  RightHandForward();
}

//Turn Robot to a gyro angle
void Minion::GyroTurnRightDegrees(int iDegrees) {
  iDegrees = iDegrees * G_Right;
  float StartAngle = 0;
  float Angle = 0;
  float diff = 0;

  mpu6050.update();
  StartAngle = mpu6050.getAngleZ();
  Angle = StartAngle;
```

```
  digitalWrite(MotorDirA, LOW);
  analogWrite(MotorPWMA, 200);
  digitalWrite(MotorDirB, HIGH);
  analogWrite(MotorPWMB, 200);

  while ((Angle - StartAngle) < iDegrees) {
    mpu6050.update();
    Angle = mpu6050.getAngleZ();
    diff = Angle - StartAngle;
  }
  RobotStop(100);

}

//Turn Robot to a gyro angle
void Minion::GyroTurnLeftDegrees(int iDegrees) {
  iDegrees = iDegrees * G_Left;
  float StartAngle = 0;
  float Angle = 0;
  float diff = 0;

  mpu6050.update();
  StartAngle = mpu6050.getAngleZ();
  Angle = StartAngle;

  digitalWrite(MotorDirA, HIGH);
  analogWrite(MotorPWMA, 200);
  digitalWrite(MotorDirB, LOW);
  analogWrite(MotorPWMB, 200);
```

```
  while ((Angle - StartAngle) > iDegrees) {
    mpu6050.update();
    Angle = mpu6050.getAngleZ();
    diff = Angle - StartAngle;
  }
  RobotStop(100);

}
```

Appendix 3. Source code of the Light Show

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
//
//  LightShow.ino - Light show for Minions performance
//
//  ver. 1.5
//  Last updated: 16/06/2019
//
//
//  14/06/2019 - Added comments
//
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////

#include <NewPing.h>

#define TRIGGER_PIN 2
#define ECHO_PIN 3
#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // UltraSonic sensor setup - pins and maximum distance

int purple[] = {255, 0, 255};
float beat = 60000 / 160;
int x = 0;
int y = 0;
int z = 0;

void setup() {
  // notes - song starts 8 seconds in and there are chorus from beats 72 to 132
```

```
  Serial.begin(9600);
  UltraSonicActivation();
  RGBCount(beat);
}

int RGBOn(int t) {
  analogWrite(9, random(0, 255));
  analogWrite(10, random(0, 255));
  analogWrite(11, random(0, 255));
  delay(t);
  analogWrite(9, 0);
  analogWrite(10, 0);
  analogWrite(11, 0);
  delay(t);
}

int ColorOn(int col[], int t) {
  analogWrite(9, col[0]);
  analogWrite(10, col[1]);
  analogWrite(11, col[2]);
  delay(t);
}

int ColorFlash(int col[], int t) {
  analogWrite(9, col[0]);
  analogWrite(10, col[1]);
  analogWrite(11, col[2]);
  delay(t);
  analogWrite(9, 0);
  analogWrite(10, 0);
  analogWrite(11, 0);
  delay(t);
}
```

```
int RGBCount(int t) {
  Serial.println("IT WORKS");
  while (x < 34 ) {
    analogWrite(9, random(0, 255));
    analogWrite(10, random(0, 255));
    delay(t);
    analogWrite(11, random(0, 255));
    analogWrite(9, 0);
    analogWrite(10, 0);
    analogWrite(11, 0);
    delay(t);
    x++;
    z++;
    Serial.println(z);
  }
  while (y < 32) {
    analogWrite(9, 255);
    analogWrite(10, 255);
    analogWrite(11, 0);
    delay(t / 2);
    analogWrite(9, 0);
    analogWrite(10, 0);
    analogWrite(11, 0);
    delay(t / 2);
    analogWrite(9, 0);
    analogWrite(10, 0);
    analogWrite(11, 255);
    delay(t / 2);
    analogWrite(9, 0);
    analogWrite(10, 0);
    analogWrite(11, 0);
    delay(t / 2);
```

```
    y++;
    z++;
    Serial.println(z);
  }
  while (z < 96) {
    analogWrite(9, random(0, 255));
    analogWrite(10, random(0, 255));
    analogWrite(11, random(0, 255));
    delay(t);
    analogWrite(9, 0);
    analogWrite(10, 0);
    analogWrite(11, 0);
    delay(t);
    z++;
    Serial.println(z);
  }
}

void UltraSonicActivation() {
  int uS = sonar.ping_cm();
  delay(50);
  while ((uS == 0) || (uS < 30)) {
    uS = sonar.ping_cm();
    Serial.print("Activation: ");
    Serial.print(uS);
    Serial.println("cm");
    delay(50);
  }
  Serial.println("Start dancing");
}

void loop() {
  // put your main code here, to run repeatedly:
```

```
  delay(100000);
}
```