Topic:-Interactive Game

CS321 Peripherals Lab

Guide:- Prof. Shivshankar B. Nair

Tushar Semwal(TA)

Team Members:- Nikit Begwani(130101055)
Rohan Gupta(130101066)
Mohit Chhajed (130101048)
Piyush Kedia(130101056)

INTERACTIVE MAZE

WINTER'15

This document contains the complete documentation of the project. It includes the guide to setup the project along with a proper documentation and code snippets which allows anyone to develop this game.

TABLE OF CONTENTS

Contents

INTERACTIVE MAZE

ABSTRACT

This project was developed as a part of our Peripherals Lab course. The topic chosen by our team was “INTERACTIVE GAME”. We have made an interactive maze, the game which we used to play in our childhood, and the ones which you can find on the top of a pencil box or a bottle. We have tried to replicate the same concept in a magnified form by developing a maze platform having two degrees of freedom and which can be controlled using an accelerometer or an AMGP (Accelerometer, Magnetic, Gyroscope and pressure) sensor.

The goal in the game is to take the metal ball located at the start point to the destination in shortest time possible using hand gesture to make the maze rotate in desired direction.

Hardware Module

Under this section we try to talk about the hardware components used and the purpose of using them. We explain about the electric and non-electric components (used for assembly).

Electronic Components used:-

1) Arduino Mega 2560:- Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. We have used this to rotate the servo motor by interpreting the data from the AMGP sensor.
2) Servo Motors(2 nos) :- A **servomotor** is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration.[1] It consists of a suitable motor coupled to a sensor for position feedback. We use the servo motor to have two degree of freedom, one along the Y-Z axis and other along the X-Z axis. Our servo motor works on the data received from the AMGP sensor.
3) IR sensors(2nos):- It measures the infrared light radiating from objects in its field of view. We are using this to sense the start moment and end moment of the game, so that we can display the time taken for completion of the game and can reset the system once one instance of the game is done.
4) LCD display :- We use an Arduino compatible LCD display to output the time taken for the game to complete.
5) AMGP sensor:- We are using Brigosha's AMGP sensor. AMGP stands for accelerometer, magnetometer, gyroscope and pressure sensor respectively. Using this sensor we get the data from user and we decide that how much and around which axis we must rotate the maze.
6) Wires:- Jumper wires were used to connect the peripherals with Arduino.

Non-Electronic Components

1) Wooden Planks: - To make the outer most frame of the game.
2) Aluminum 'L' shaped strips – To make the base of the game. The 2plane rotating base.
3) Aluminum 'L' angles – Used to join the edges of the outer frame.

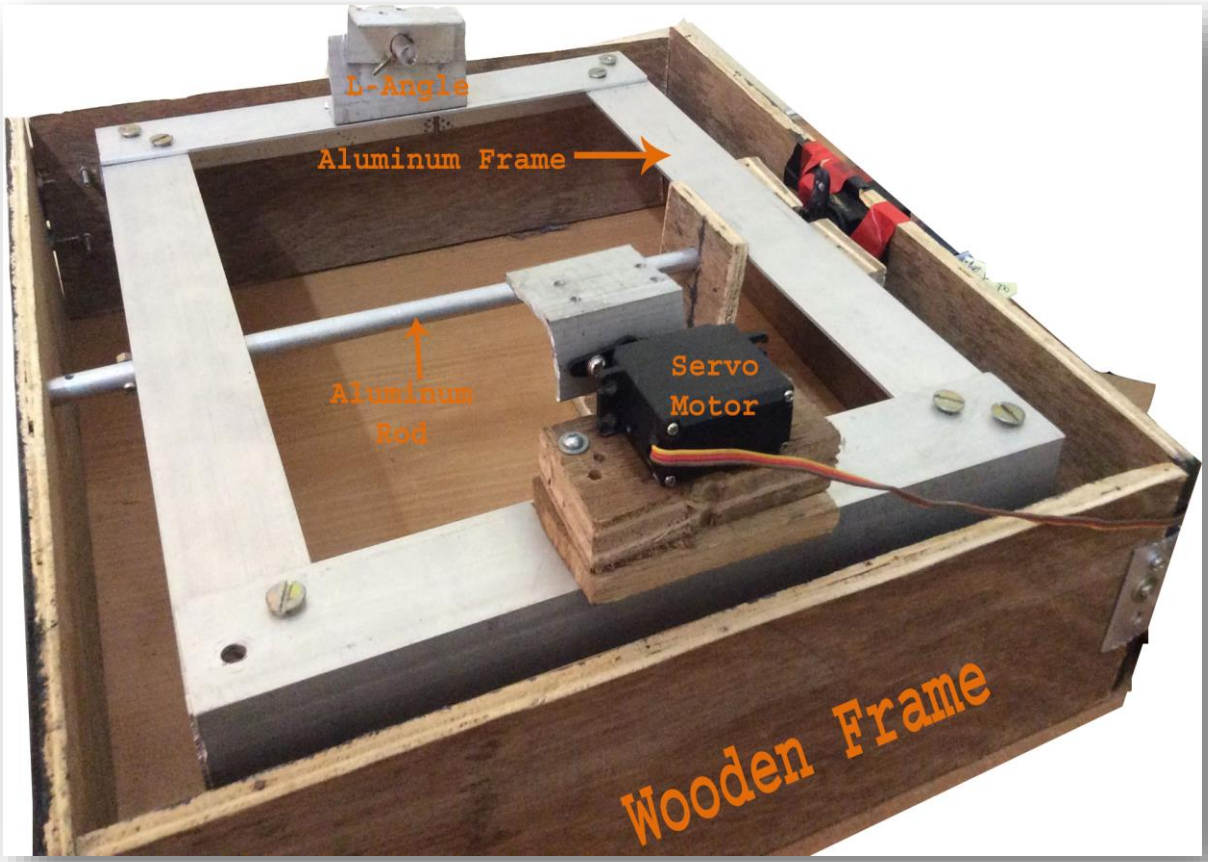
4) Aluminum rods: - These were used to give support to the rotating base and to provide it an axis of rotation for free movement.
5) Cardboard: - The maze is made of cardboard so that we can have a replaceable maze which helps users to play on different difficulty levels.
6) Screws, nuts and bolts: - to make the assembly strong and durable.
7) Metal ball: - To play the game.

TOOLS USED FOR MAKING THE ASSEMBLY

1) BOSCH electric drill and drill bits
2) Saw
3) Hammer
4) Screw driver and screws
5) Fevicol and fevikwik.
6) Both sided tapes and normal tapes.

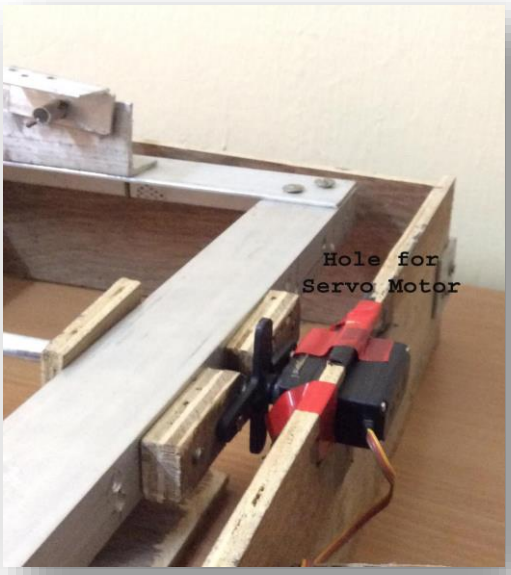
Assembly

The assembly was made by us right from scratch, we prepared the design first on pen and paper before implementing it. The assembly consists of three rectangular frames, the outer frame made of wood and jointed at the edges by “L” angles. We made two holes in this one of size of the servo motor and other a small round hole at exact opposite end to insert the aluminum rod(for free movement).



After that we proceeded with making the aluminum frame for rotation in one plane, we made this of aluminum because we wanted it to be light and at the same time we wanted it to be strong and firm, the frame is composed of connected aluminum L shaped strips This frame also has two small circular holes for passage of rod, it is also provided with an

inverted T shaped wooden support, so that the torque of servo is protected when the game is in off condition so that it stay balanced at center. On two opposite end of the frame, we have attached objects which support the next plank, On one side, we have attached two L shaped angles connected to each other via a hole and small aluminum rod, for free rotation in other direction and on the other end, we have attached two wooden blocks to raise the height of servo motor.



Servo Fitting

Maze

The next layer is the maze game itself, we created it using thick cardboard sheet, it lies on the top of L angles previously described, thus leading to completion of the assembly.

Software Module

Arduino and its Libraries

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

To learn how to use an Arduino, one can visit the link here

Libraries used

1) Wire.h - This library allows us to communicate with I2C / TWI devices. On the Arduino boards with the R3 layout (1.0 pinout), the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. The Arduino Mega 2560 has the SDA pin at pin number 20 and SCL pin at pin number 21. We use these pins for the AMGP sensor.

2) Servo.h - This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

3) I2Cdev.h - The **I2C Device Library** is a collection of uniform and well-documented classes to provide simple and intuitive interfaces to an ever-growing collection of I2C devices. Each device is built to make use of the generic i2cdev code, which abstracts the I2C bit-level and byte-level communication away from each specific device class, making it easy to keep the device code clean while providing a simple way to modify just one class to port the I2C communication code onto different platforms. We use this library to interface with the AMGP sensor.

4) BMA150.h – We use this library to interface with the accelerometer of the AMGP sensor, this library consists of all the functions related to the accelerometer data in all the axis, we call initialize to initialize the value in all the three axis and getacceleration (with three parameters) to get the value of acceleration in all the three axis.

5) ITG3200.h - We use this library to interface with the gyroscope of the AMGP sensor, this library consists of all the functions related to the gyroscope data in all the axis, we call initialize to initialize the value in all the three axis and getrotation (with three parameters) to get the value of angular velocity in all the three axis.

We declare the objects of each type like servo, accel and gyro as shown in the code to access the functions of the library. The raw data at times are not sufficient to provide a smooth data to transfer to the servo motor, so we can use various types of filter to get the desired values. The well known filtering measures known are:-

1. PID(Proportional Integral and Differential)
2. Complimentary Filter
3. Kalman Filter.

We can easily find resources on internet which explains about these filtering techniques.

Checking Values

The Arduino software provides a facility to see the values by printing them to the serial monitor , we can print the values and using the hardware components, we can analyse the values given by them.

Code

Below are the code snippets which we have implemented, the code is commented well which helps in understanding it better.

```
#include <ITG3200.h>
#include <I2Cdev.h>
#include <HMC5883L.h>
#include <BMA150.h>
#include <Servo.h>
#include <Wire.h>

BMA150 accel;       // object for accelerometer
ITG3200 gyro(0x69);   // object for gyroscope
int temp, temp2;      // temporary objects to provide data to servo motor

int16_t ax, ay, az;   //axis wise value of accelerometer
int16_t gx, gy, gz;   //axis wise value of gyroscope

Servo myservo, myservo2;    //servo objects
uint32_t timer;

double  pitch, roll, yaw, pitch_acc, roll_acc, yaw_acc; //pitch yaw and roll are the angles of projection on
                                                        // X, Y, Z axis
const int button_pin = 36;

void setup() {
  pinMode(3, INPUT);          // Pin configuration
  pinMode(button_pin, INPUT);
  myservo2.attach(11);    //servo pin defined for one mode of rotation
  myservo.attach(9);      //servo pin defined for other mode of rotation
  // join I2C bus (I2Cdev library doesn't do this automatically)

  Wire.begin();
  Serial.begin(9600);

  // Initialize devices
  //Serial.println("Initializing I2C devices...");
  accel.initialize();
  gyro.initialize();

  // verify connection
  //Serial.println("Testing device connections...");
  //Serial.println(accel.testConnection() ? "BMA150 connection successful" : "BMA150 connection failed");
  //Serial.println(compass.testConnection() ? "HMC5883L connection successful" : "HMC5883L connection failed");
  //Serial.println(gyro.testConnection() ? "ITG3200 connection successful" : "ITG3200 connection failed");
```

```
  accel.getAcceleration(&ax, &ay, &az);       //getting values from the device
  gyro.getRotation(&gx, &gy, &gz);

  pitch_acc = 0;
  roll_acc = 0;
  yaw_acc = 0;
  pitch = pitch_acc;
  roll = roll_acc;
  yaw = yaw_acc;
  timer = micros();        //Initialize timer
}

void loop() {
//  if (digitalRead(button_pin)) {
    // read raw gyro measurements from device
    if (digitalRead(3) == HIGH)
        Serial.print("1\n");
    else
        Serial.print("0\n");
    //Serial.print("\n");
    accel.getAcceleration(&ax, &ay, &az);   //getting acceleration value continuously

                          //---------COMPLIMENTARY FILTER---------------//
     /*gyro.getRotation(&gx, &gy, &gz);     //getting gyroscope values continuously

    double dt = (double)(micros() - timer) / 1000000; // Calculate delta time
    timer = micros();

    pitch += ((double)gx / 14.375) * dt;            //Raw gyro data 14.375 is sensitivity value of gyroscope used
    roll += ((double)gy / 14.375) * dt;
    yaw += ((double)gz / 14.375) * dt;

    int forceMagnitudeApprox = abs(ax) + abs(ay) + abs(az);
    if (forceMagnitudeApprox > 200 && forceMagnitudeApprox < 500)
    {
      pitch_acc = atan2((double)az, (double)ay) * 180 / M_PI;          //Raw accel data
      roll_acc = atan2((double)ax, (double)az) * 180 / M_PI;
      yaw_acc = atan2((double)ay, (double)ax) * 180 / M_PI;

      pitch = pitch * 0.98 + pitch_acc * 0.02;
      roll = roll * 0.98 + roll_acc * 0.02;
      yaw = yaw * 0.98 + yaw_acc * 0.02;
    }*/

                        //---------COMPLIMENTARY FILTER ENDS---------------//
```

```
   temp = map(ax%360, -230, 230, 69, 88);     // scale it to use it with the servo (value between 0 and 180)
   temp2 = map(ay%360, 230, -230, 85, 110);     // scale it to use it with the servo (  ue between 0 and 180)
    myservo.write(temp);
     myservo2.write(temp2);

    //Serial.println(forceMagnitudeApprox);
    // display tab-separated accel x/y/z values
    //Serial.print("accel:\t");            //printing values in serial monitor to test
    //Serial.print(ax); Serial.print("\t");
    //Serial.print(ay); Serial.print("\t");
    //Serial.println(az);

    delay(10);
  }
```

Limitations

We made an effort to keep the project as free from errors and limitations as possible. We consider the following points as the limitations of our project

1) The servo motor placed inside the cutting of the wooden frame, may cause and wear and tear to the frame if the user turns the AMGP sensor at extremely fast rate, causing the servo motor to change the angles rapidly.
2) When the game is not in use, there is a pressure on the servo motor to hold the aluminum frame straight which may hamper the motor in long run.
3) At present, we have just provided with a single maze, we can provide more such maze so that a person can change the maze and can experience various difficulty levels.
4) The complimentary filter renders the data slow and increases or decreases the data linearly with a bit of calculations. This leads to slow movement of servo motor, at present we are getting good and smooth results with the accelerometer data only but we feel the results can be improved more if we can render the data fast from the complimentary filter.

Resources

1) https://www.arduino.cc/en/Reference/HomePage
2) http://www.students.brigosha.com/embedded/assets/downloads/notes/amgp.pdf
3) Video link