Week 2 Report - Tinkering Lab – PG20 – Project: Air Mouse

Members and corresponding contributions:

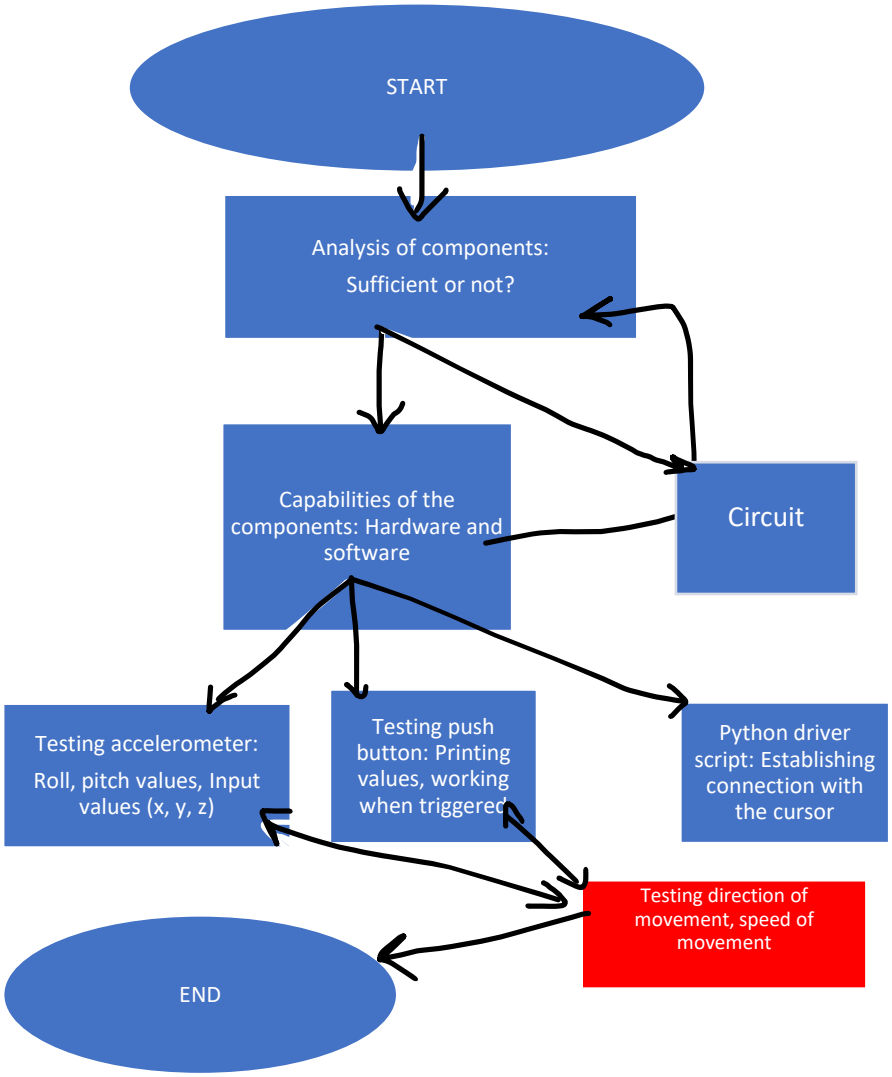
1. Karthik Kumar Pradeep (2022EPB1234)
    - *Contribution*: Programmed the accelerometer/Arduino and Python driver script. Undertook testing and troubleshooting to handle hardware and software issues
2. Khushi Tapariya (2022EPB1235)
3. Manasvi Swarnkar (2022EPB1237)
    - *Contribution of Members 2 and 3*: Surveyed the capabilities of the components used in the project so far and how they can be utilized for the project.
4. Mayank Sharma (2022EPB1238)
    - *Contribution*: Surveyed various libraries in Arduino IDE and their possible usage for the project keeping in mind the restrictions of existing components
5. Nilesh Kumar (2022EPB1239)
6. Nishant Kumar (2022EPB1240)
    - *Contribution of Members 5 and 6*: Worked towards finding solutions for hardware limitations. Looked for alternative ways to use the hardware for the project.

Progress until this week

ATMega328P microcontroller was used along with ADXL335 accelerometer and push buttons at first to implement the air mouse directly i.e., direct readout and use of x, y, and z values as determined by the accelerometer. However, there were many issues with the working of the components, as there was no proper communication between the microcontroller and the components

Tasks taken up in Week 2

- Workflow diagram:



- Objective: To first build towards a stage at which changing the orientation of the accelerometer results in the appropriate motion of the computer screen cursor.

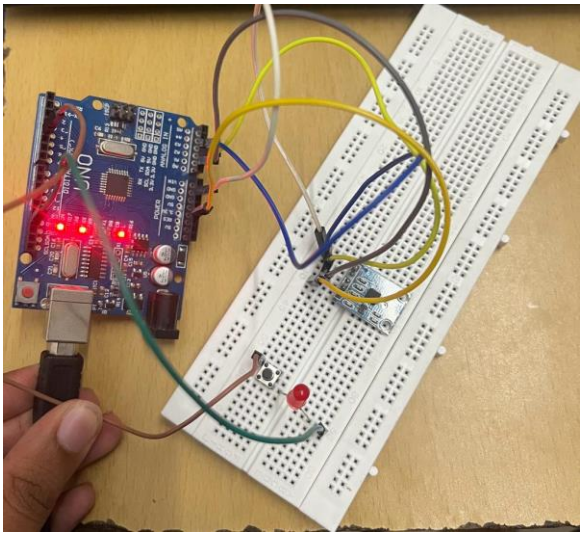
- Problem Statement: Test the accelerometer alone to see how it works and to observe the changes in its parameters. Establish a connection between the screen cursor and Arduino; to interface it with the accelerometer.

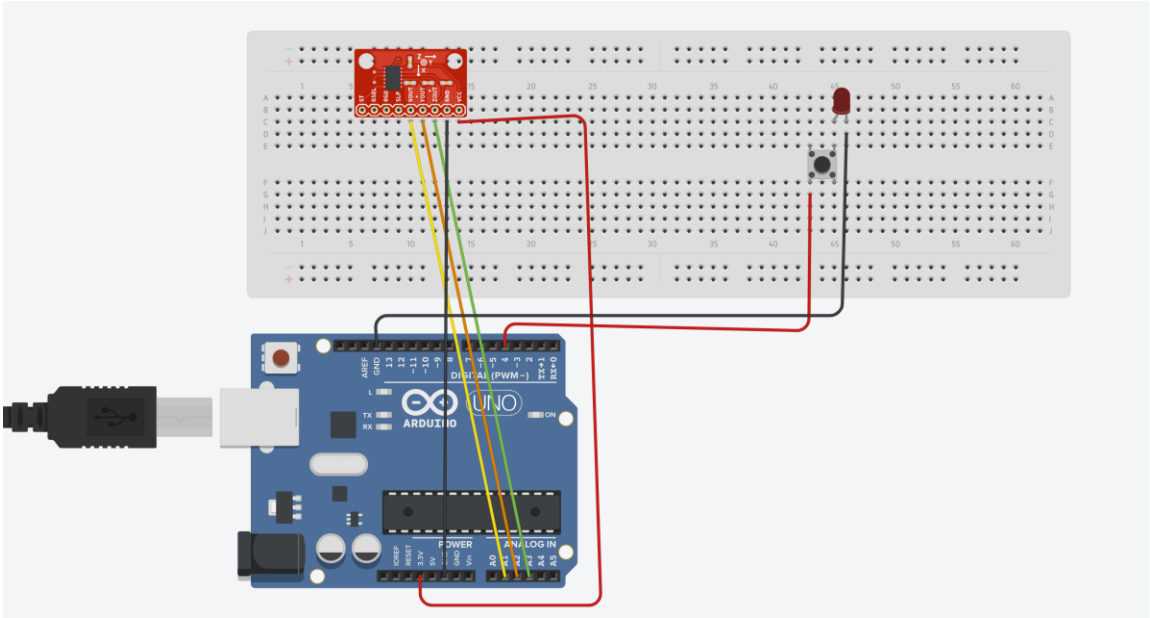
- Components used: Arduino UNO, ADXL335 Accelerometer, Push button, LED, Jumper wires, Breadboard.

- Hardware: We connected the analog pins of Arduino to x-out, y-out, z-out pins of the accelerometer respectively. This effectively reads the acceleration of the accelerometer sensor. Push button was connected to digital pin 4 and an LED was connected as well along with the grounding.

- Working: By changing the orientation of the breadboard (essentially the accelerometer sensor) in space (air), we can change the roll and pitch values of the sensor. These roll and pitch values, just like how they are important for a pilot flying an airplane, would help us move the cursor on the screen. The push button acts as a trigger, which when pressed helps send a signal to the cursor to move according to the real-time roll and pitch values read out by the accelerometer.

- Circuit: ***Note that the ADXL335 used has only 5 pins unlike the below Tinkercad diagram***





- <u>Software</u>: The Arduino code programs the accelerometer to read out the x, y, and z values and utilizes it to calculate the roll and pitch values and display them on the

serial monitor. Appropriate scaling is done at various stages. A Python driver script is also run, which establishes a connection between the Arduino and the screen cursor.

- Arduino code Logic
  - Math header file is included as the atan2 function (arc tan) is used to calculate roll and pitch values. Declarations are made for the variables corresponding to connections to different pins. Serial communication baud rate is set and for the push button, pull-up input is specified. The push button has to read whether it is pushed (high) or not (low) – pullup is used in order to avoid the usage of a resistor in the circuit, as it can be compensated by the Arduino board
  - Voltage values of x-out, y-out, z-out pins which are read out are converted to acceleration values and scaled in terms of units of g (acceleration due to gravity). Roll and pitch are then calculated using these x, y, z values and the atan2() function.
  - The roll and pitch values, from the range of its possible values are mapped to a range of values (-50, 50) which the cursor moves according to. These new mapped values are say, X and Y.
  - When the pushbutton is pressed, the corresponding mapped roll and pitch value (X, Y) get printed on the serial monitor
  - For observing the parameters of the accelerometer lines 25 to 33, and 53 to 58 were uncommented. However, as we only wish to print the X, Y values necessary for movement of cursor, these lines are finally commented.
- Python Driver Script:
  - Libraries: serial – helps use the output in the serial monitor of specific communication port that the Arduino is connected to. Pyautogui – controls the mouse cursor through the code.
  - X, Y values are individually readout from serial monitor and pyautogui then accordingly moves the cursor according to it.
- Code snippets:
- Arduino code:
-

```
#define BUTTON_PIN 4

#include <math.h>
const int x_out = A1; /* connect x_out of adxl335 to A1 of UNO board */
const int y_out = A2; /* connect y_out of adxl335 to A2 of UNO board */
const int z_out = A3; /* connect z_out of adxl335 to A3 of UNO board */

void setup()
{
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
}

void loop()
{
  byte buttonState = digitalRead(BUTTON_PIN);
  int x_adc_value, y_adc_value, z_adc_value;
  double x_g_value, y_g_value, z_g_value;
  double roll, pitch, yaw;
  x_adc_value = analogRead(x_out); /* Digital value of voltage on x_out pin */
  y_adc_value = analogRead(y_out); /* Digital value of voltage on y_out pin */
  z_adc_value = analogRead(z_out); /* Digital value of voltage on z_out pin */

  /*
  Serial.print("x = ");
  Serial.print(x_adc_value);
```

-
```
  Serial.print("x = ");
  Serial.print(x_adc_value);
  Serial.print("\t\t");
  Serial.print("y = ");
  Serial.print(y_adc_value);
  Serial.print("\t\t");
  Serial.print("z = ");
  Serial.print(z_adc_value);
  Serial.print("\t\t");
  */
  //delay(100);

  x_g_value = ( ( ( (double)(x_adc_value * 5)/1024) - 1.65 ) / 0.330 ); /* Acceleration in x-direction in g units */
  y_g_value = ( ( ( (double)(y_adc_value * 5)/1024) - 1.65 ) / 0.330 ); /* Acceleration in y-direction in g units */
  z_g_value = ( ( ( (double)(z_adc_value * 5)/1024) - 1.80 ) / 0.330 ); /* Acceleration in z-direction in g units */

  roll = ( ( (atan2(y_g_value,z_g_value) * 180) / 3.14 ) + 180 ); /* Formula for roll */
  pitch = ( ( (atan2(z_g_value,x_g_value) * 180) / 3.14 ) + 180 ); /* Formula for pitch */
  //yaw = ( ( (atan2(x_g_value,y_g_value) * 180) / 3.14 ) + 180 ); /* Formula for yaw */
  /* Not possible to measure yaw using accelerometer. Gyroscope must be used if yaw is also required */

  // Observed (favorable): Roll varies roughly from 81 to 283
  // Pitch from say 85 to 280

  int x = map(roll, 1.00, 355.0, -50.0, 50.0);
  int y = map(pitch, 85.0, 280.0, -50.0, 50.0);
```
-
```
  int x = map(roll, 1.00, 355.0, -50.0, 50.0);
  int y = map(pitch, 85.0, 280.0, -50.0, 50.0);

/*
  Serial.print("Roll = ");
  Serial.print(roll);
  Serial.print("\t");
  Serial.print("Pitch = ");
  Serial.print(pitch);
  Serial.print("\n");
*/
  if (buttonState == LOW)
  {
     //Serial.println("Button is pressed");
     Serial.print(x);
     Serial.print(",");
     Serial.println(y);
     //Serial.println("OK");
  }
  else
  {
     //Serial.println("Button is not pressed");
  }
  delay(100);
}
```
- Python driver script:
-
```
import serial
import pyautogui

ser = serial.Serial('COM6', 9600)

while True:
    try:
        data = ser.readline().decode().strip().split(',')
        x, y = map(int, data)
        pyautogui.move(x, y)
    except ValueError:
        pass
```

```
PS C:\Python\Scientific> & "C:/Program Files/Python310/python.exe" c:/Python/Scientific/tl_am.py
```
- Output: ***X, Y (pair of values) are only printed when the push button is pressed (for 2nd and 3rd images)***
-
```
x = 336   y = 345   z = 403   Roll = 191.65   Pitch = 273.25
x = 336   y = 345   z = 403   Roll = 191.65   Pitch = 273.25
x = 337   y = 345   z = 403   Roll = 191.65   Pitch = 271.58
x = 337   y = 345   z = 402   Roll = 191.99   Pitch = 271.63
x = 337   y = 345   z = 403   Roll = 191.65   Pitch = 271.58
x = 336   y = 345   z = 403   Roll = 191.65   Pitch = 273.25
x = 337   y = 345   z = 403   Roll = 191.65   Pitch = 271.58
x = 337   y = 345   z = 403   Roll = 191.65   Pitch = 271.58
x = 336   y = 345   z = 402   Roll = 191.99   Pitch = 273.34
x = 336   y = 346   z = 402   Roll = 193.62   Pitch = 273.34
x = 337   y = 346   z = 403   Roll = 193.24   Pitch = 271.58
x = 337   y = 345   z = 402   Roll = 191.99   Pitch = 271.63
```

- 

- Result: The program prints without any issue the x, y, z values as well as the pitch and roll values, confirming proper working of accelerometer, especially while moving it through the air. We also managed to successfully interface the circuit with the computer cursor which moved in a definite pattern only when the trigger was given. However, movement of cursor and accelerometer is not in the same sense currently

Challenges faced and Future work

- Challenges:
  - There were lot of software limitations that the Arduino Uno posed in this work, as it was not compatible with many of the essential libraries essential for interfacing the hardware with the cursor.
  - The ADXL335 accelerometer couldn't calculate yaw – this could pose limitations with regards to the scalability of this method of controlling the cursor in say, 3d movement
  - Main challenge: Currently the cursor moves definitely, but vertical/horizontal motion of the air mouse doesn't result in the same vertical/horizontal motion of the cursor but rather a similar motion along a diagonal
- Solutions (implemented/considered for future work):
  - The appropriate usage of Python Driver script proved to be useful in bypassing the communication issues of the Arduino Uno.
  - We could use an additional push button and see how to use roll and pitch values to compensate for yaw.
  - Main Plan: Change the mapping of roll and pitch values based on multiple trials and observation to synchronize the air mouse motion with that of the cursor.

- Future work: Improvise on the code for X, Y values to synchronize the air mouse motion, make it smoother, in the right direction. We will also try to implement right and left click push buttons.

Summary

Thus far, we have been able to successfully interface our circuit, and thus the air mouse (the breadboard itself) with the cursor on the screen. We have tested the accelerometer and seen how the roll and pitch values change. We utilized the real-time roll and pitch values to control the cursor. The cursor is moving in a corresponding way. However, we are yet to synchronize the air mouse with the cursor to have motion in same direction.