Week 3 Report – Tinkering Lab – PG20 – Project: Air Mouse

Members and corresponding contributions

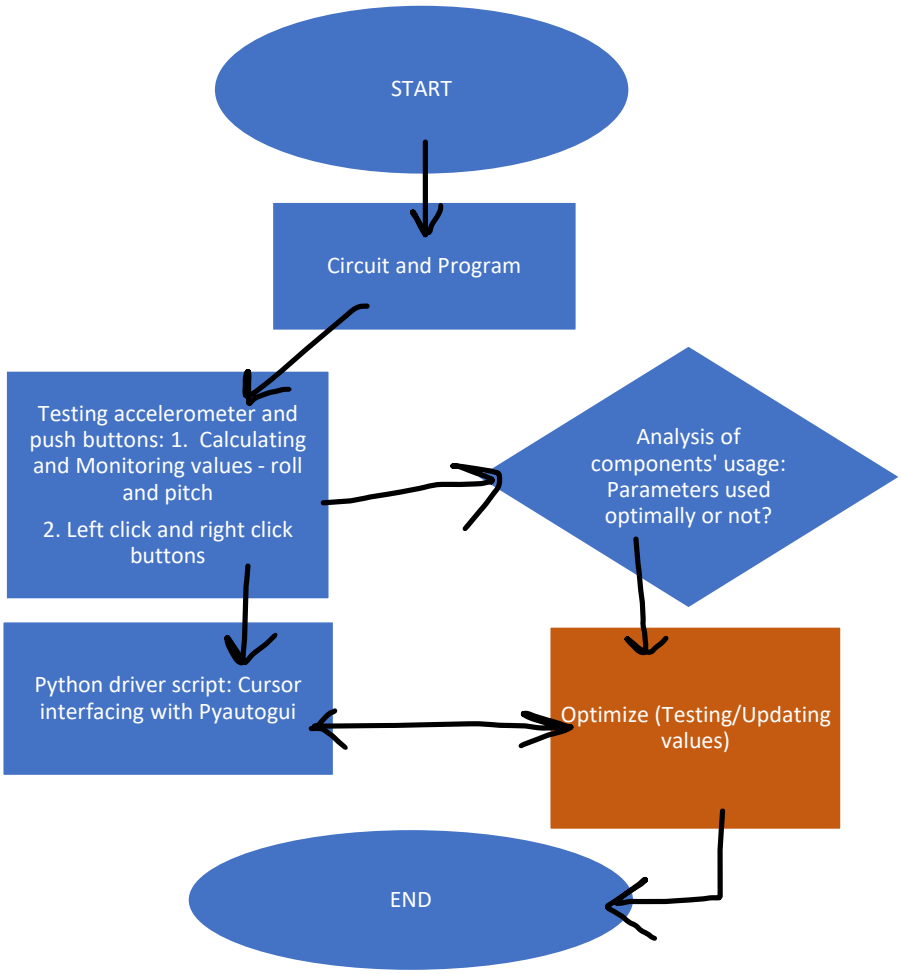
1. Karthik Kumar Pradeep (2022EPB1234)
    - *Contribution*: Programmed the accelerometer/Arduino and Python driver script. Undertook testing and troubleshooting to handle hardware and software issues
2. Khushi Tapariya (2022EPB1235)
3. Manasvi Swarnkar (2022EPB1237)
    - *Contribution of Members 2 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)
5. Nilesh Kumar (2022EPB1239)
6. Nishant Kumar (2022EPB1240)
    - *Contribution of Members 4, 5, and 6:* Surveyed the libraries used in the project. Worked towards finding solutions for hardware limitations faced in the project.

Progress until this week

Arduino Uno was used along with the accelerometer ADXL335 and push button which acts as a trigger. We have been able to establish a connection between the screen cursor and the Arduino. Motion of the accelerometer in the air, while trigger is used, caused a regular movement of the cursor. However, the movement of the two were not synchronized in the same direction.

Tasks taken up in Week 3

- Workflow diagram:



- Objective: To build towards a functional air mouse, i.e., one that has all the capabilities of a typical mouse and works desirably, in terms of cursor movement and mouse clicks.

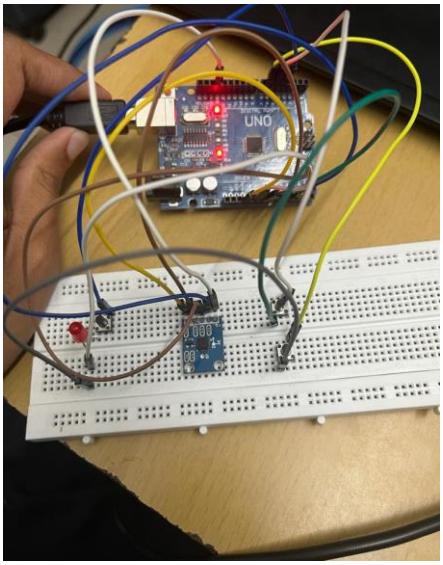
- Problem Statement: To test how cursor movement depends on various parameters such as roll, pitch, and corresponding mapped values and to make changes that give appropriate motion of the cursor corresponding to the motion of the air mouse. Include push buttons for left and right clicks of the air mouse.

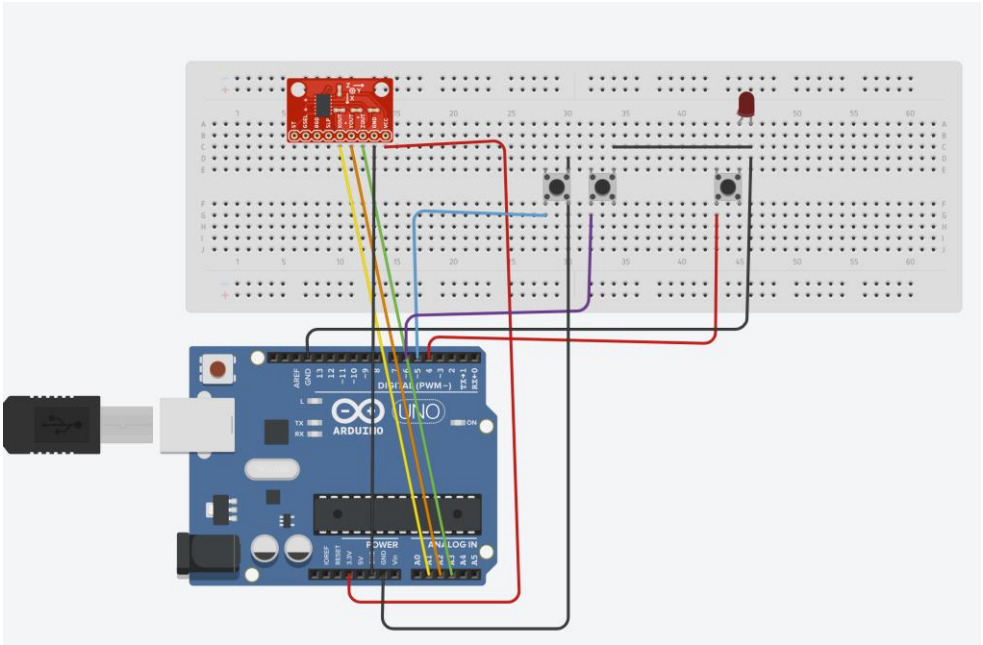
- Components Used: Arduino UNO (with its USB cable), ADXL335 Accelerometer, 3 Push buttons, LED, Jumper Wires, Breadboard.

- Hardware: There were no major changes to the hardware, as our task this week mostly involved appropriate tuning of the air mouse, which was only possible through the program controlling Arduino. Using the existing circuit, this week we included 2 push buttons aimed to function as left and right click buttons for the air mouse. They were connected to digital pins 5, 6 and grounded accordingly.

- Working: Just as we implemented last week, changing the orientation of the breadboard in space (air), causes changes in roll and pitch values, which are the main parameters responsible for cursor movement. We have added 2 push buttons, each of which when pressed, sends a signal to the cursor to execute a left or right click respectively. There was no need to use resistors, as we made the push buttons to utilize the in-built pull up resistor of the Arduino board through the Arduino code (INPUT_PULLUP)

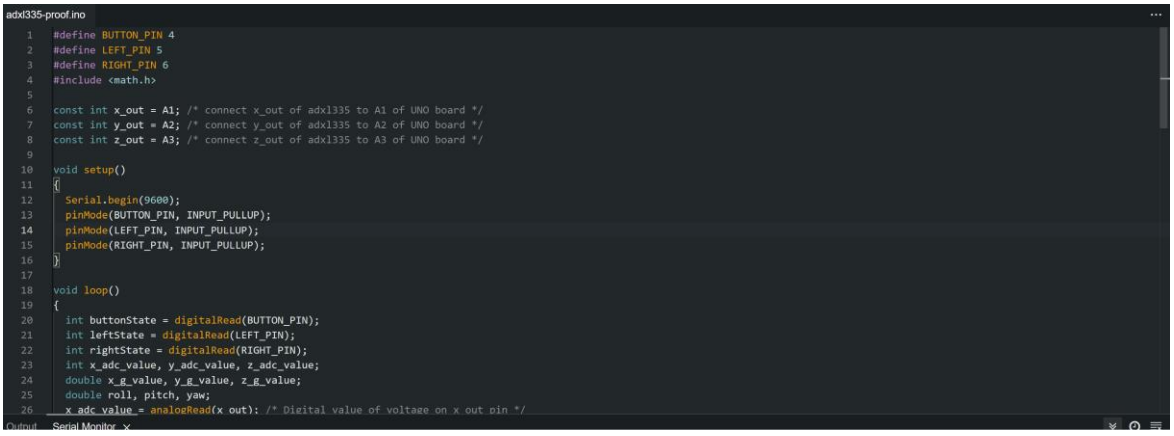
- Circuit: ***Note that the ADXL335 used has only 5 pins unlike the below Tinkercad diagram***



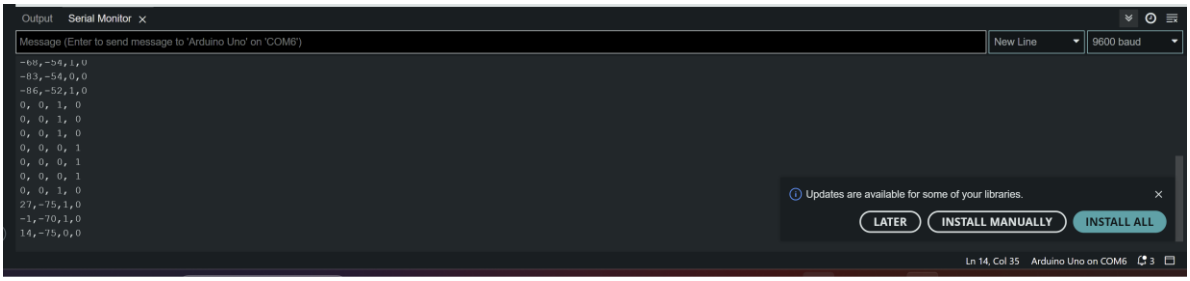


- Software: Just as has been implemented so far, the Arduino code programs the accelerometer to read out x, y, and z values which are then used to find roll and pitch

values. The appropriately mapped roll and pitch values (for cursor motion) are printed onto the serial monitor.

- ***This week's additions***: One of the most important additional lines in the Arduino code was used to change the formula for pitch value calculated. Code was also written to program the Arduino to recognize whether the push buttons for left/right click are pressed or not. The Python driver script correspondingly executes a left/right click
- Additional lines in Arduino code:
  - Lines 2, 3, 14, 15 declared the push buttons connected to digital pins 4 and 5 as input using pull up resistor. Lines 20, 21 check whether these push buttons are in HIGH or LOW state
  - Line 48 calculates the value of pitch based on readout values of the accelerometer. Upon more careful understanding of the mechanics and readout values, we corrected the pitch formula, which included the previously used arctan function (atan2()) along with sq() and sqrt() functions
  - Lines 56, 57: We changed the mapped roll and pitch values to vary from -180 to 180 degrees, while choosing appropriate maximum and minimum values of roll and pitch.
  - Between Lines 67-95: This piece of code prints out the state of left/right click push buttons onto the serial monitor along with the mapped roll and pitch values.
- Additional lines in Python driver script:
  - Through Line 11, the Python driver script reads out the Mapped roll, pitch values along with state of left/right click buttons.
  - Lines 12, 17: Mapped roll and pitch values are used to offset the present position of cursor; state of left/right buttons decide whether click functions are called or not.
- <u>Code Snippets</u>:
- Arduino code:
- 

-
```
  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("\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(-x_g_value,sqrt(sq(y_g_value) + sq(z_g_value))) * 180) / 3.14 ) + 180 ); /* Formula for pitch */
  // 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 */
```
-
```
  // 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 45 to 300
  // Pitch from say 90 to 250

  int x = map(roll, 45.00, 300.0, -180.0, 180.0);
  int y = map(pitch, 90.0, 250.0, -180.0, 180.0);

 /*
  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.print(y);
      //Serial.println("OK");
```
-
```
      //Serial.println("OK");

      Serial.print(",");
      if (leftState == LOW)
        Serial.print(1);
      else
        Serial.print(0);

      Serial.print(",");
      if (rightState == LOW)
        Serial.println(1);
      else
        Serial.println(0);
  }
  else
  {
      if (leftState == LOW && rightState == LOW)
        Serial.println("0, 0, 1, 1");
      if (leftState == HIGH && rightState == LOW)
        Serial.println("0, 0, 0, 1");
      if (leftState == LOW && rightState == HIGH)
        Serial.println("0, 0, 1, 0");
      //Serial.println("Button is not pressed");
  }
  delay(120);
}
```
- Python driver script:
```
import serial
import pyautogui

# y is the mapped pitch val, x is the mapped roll val

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

while True:
    try:
        data = ser.readline().decode().strip().split(',')
        x, y, l, r = map(int, data)
        pyautogui.move(x, y)
        if (l == 1):
            pyautogui.click()
        elif (r == 1):
            pyautogui.rightClick()
        print(f"{x}, {y}")
    except ValueError:
        pass
```
- Output: ***Values are printed only when push buttons are used; they change when the orientation of the air mouse changes. The last 2 binary values are for left and right clicks respectively.***

- 

- Result: The Arduino program prints out the values of mapped roll and pitch values without any issue, and also specifies whether left/right click buttons are pressed (1) or not (0). We observed that the changing roll and pitch values by moving the air mouse in various directions resulted in the desirable motion of the cursor on the screen in the corresponding direction in accordance with the mapped roll and pitch, while using the trigger. The left and right click buttons are also successfully working. Thus, we have made a fully functional air mouse with the limited capabilities of the existing components. However, there is scope for improvement concerning user experience.

Challenges faced and future work

- The air mouse is now functional. For improvements, there are still some major challenges – concerning the limitations of the hardware used.
- Challenges:
  - The air mouse made does not perform as efficiently as a typical mouse would. For instance, the motion of the cursor is not smooth and is in 'discrete' steps. Also, it is difficult to increase the speed of the cursor movement without compromising on the control over the cursor.
  - Simultaneous clicking and motion of the cursor is cumbersome with the current air mouse.
- Solutions (for future work):
  - Inclusion of another push button for 'dragging' (clicking, holding, and moving the cursor)
  - Changes in the mapped roll and pitch values to get more efficient results.
  - Utilizing the functions in the Pyautogui library more rigorously.

- Future work: Improvise on the code by using the most appropriate mapping of roll and pitch values. To utilize the relevant capabilities of the Pyautogui library fully. Implement push button for 'dragging'

Summary

With the work done until this week (Week 3), we have successfully managed to implement a functional air mouse. The work is based on the idea of roll and pitch values of the accelerometer motion in space. We have tested the air mouse and concluded that it can perform various tasks that a typical mouse can do satisfactorily (Motion and clicking) but not yet in a way more efficient than a typical mouse. We would like to now focus on improving

the user experience and aim towards improving it so that it could be preferred over a typical mouse.