Flex Sensor Project

This project uses a flex sensor to describe the current tension of the sensor using the Arduino Uno board.

Initial goal:

We thought that using the flex sensor's measurements we can calculate an approximate degree of its bending. This however failed, due to the inconsistency of the bending. The tension between its endpoints caused the middle of the sensor to bend in different ways at the same measurement time. This caused that we could not calibrate the sensor to display degrees.

Second goal:

After considering the flex sensor's circumstances we thought a more descriptive value on the sensor's current bend would be more appropriate. This has been achieved by taking measurement points, mapping them to values on a scale between 0 to 1, and using a polynomial function to blend between them.

We marked fully bent as 1.0, strongly bent as 0.8, medium bent as 0.5 and slightly bent as 0.3. We then searched appropriate measurements for these values and jump between linear interpolations based on the current measurements. The value between 0 and 500 was mapped from 0.0 to 0.3, and from 500 to 650 was mapped from 0.5 to 0.8 and from 650 to 850 is mapped from 0.5 to 0.8 and from 850 to 990 is mapped from 0.8 to 1.0. This simple solution proved to be sufficient and functional.

Later we thought that this project could be used as a peripheral to the computer, so we have implemented a mouse click functionality to the program, and thanks to the mapping we do not have double clicks, or any miss inputs.

Below is documentation for the code:
The project uses 3 code files:

- v1.ino is the code that runs on the Arduino microcomputer.
- v1.py is the python code that runs on the local computer that processes the data the Arduino provides.
- DegreeConverter.py is a module in python that converts the data into usable measurements.

v1.ino

This Arduino sketch reads an analog input from a flex sensor connected to pin A0 and prints the raw sensor value to the Serial Monitor. It operates in a continuous loop, updating the value every 100 milliseconds.

- The analog value from the flex sensor ranges from 0 to 1023, representing the resistance change of the sensor as it bends.
- This sketch provides a basic framework for further development, such as mapping the sensor value to specific ranges or implementing additional functionality based on the sensor readings.

v1.py

This script interfaces with a serial device to read data and simulate mouse clicks based on processed input values. It utilizes the pyautogui library for mouse interactions and a custom scaling function from the DegreeConverter module.

Code Explanation

Imports

serial: Facilitates reading from the serial port.

polynomial: Custom function for scaling data values.

pyautogui: Allows for mouse click simulation.

Serial Port Initialization

python Copy code

```python
ser = serial.Serial("COM4", baudrate=9600, timeout=1)
```

Initializes the serial connection on the specified COM port with a baud rate of 9600.

Sample Initialization

python Copy code

```python
sample_size = 1
sample = [0 for i in range(sample_size)]
current_index = 0
dx = 0
prev_x = None

click = 0.3
```

Defines the sample size, initializes a list to store samples, and sets up variables for tracking changes and previous values.

Main Loop

python Copy code

```python
while True:
    data = ser.readline()
    data = str(data)[2:-5]
```

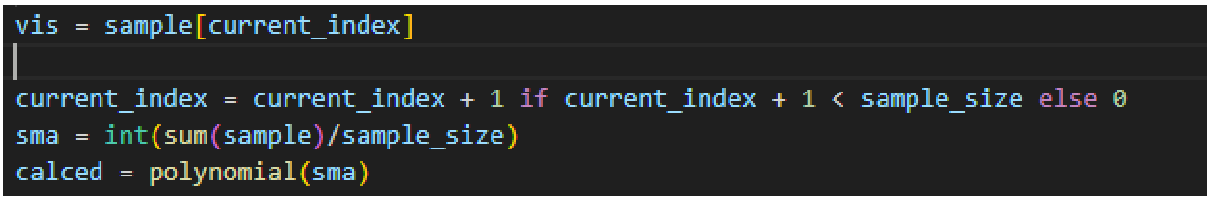
Enters an infinite loop to continuously read data from the serial port and processes it.

Data Handling

```python
try:
    sample[current_index] = int(data)
except:
    sample[current_index] = 0
```

Attempts to convert the read data to an integer, defaulting to 0 if unsuccessful.

Calculating Values



```python
vis = sample[current_index]

current_index = current_index + 1 if current_index + 1 < sample_size else 0
sma = int(sum(sample)/sample_size)
calced = polynomial(sma)
```

Updates the current sample index, calculates the simple moving average, and applies the custom scaling function.

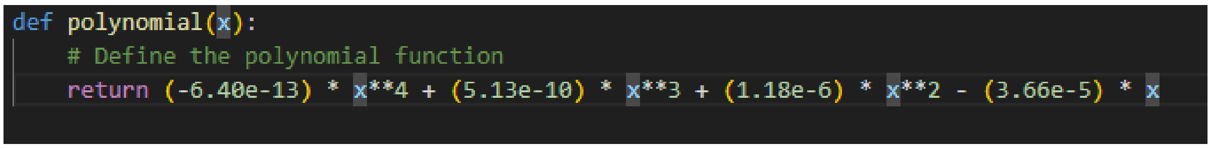
Output

```python
print(f"Value: {vis} Calculated: {calced}")
```

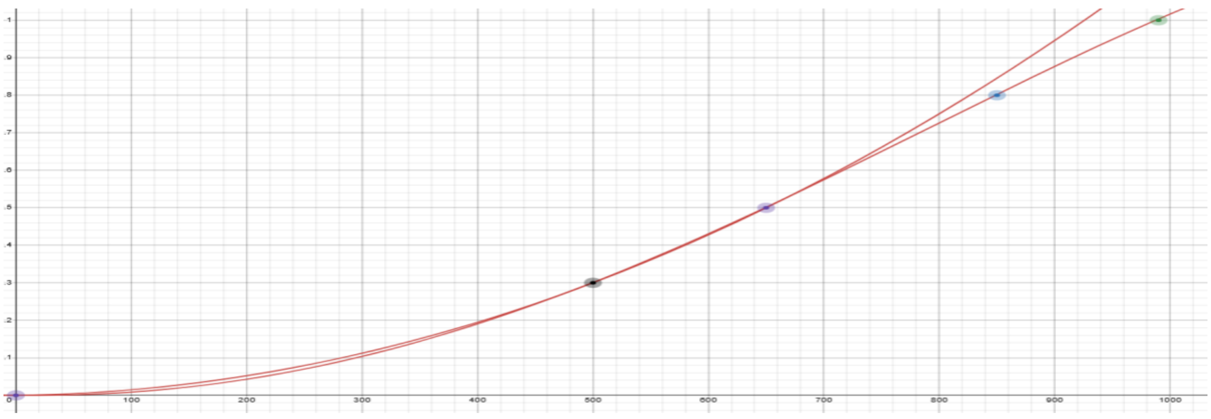
Outputs the current raw and calculated values for monitoring.

DegreeConverter.py

We used a polynomial function to scale an exponential function to be linear.



```python
def polynomial(x):
    # Define the polynomial function
    return (-6.40e-13) * x**4 + (5.13e-10) * x**3 + (1.18e-6) * x**2 - (3.66e-5) * x
```



Video link