COMPENG 2DX3: Microprocessor Systems Project

3D Spatial Mapping using Time of Flight Sensor

Sarah Ahmed - L02
Instructors: Yaser Haddara, Sharukh Athar, Thomas Doyle
Submission Date: April 17, 2024

Device Overview

This is a smart portable and affordable LIDAR system designed to create a 2D mapping of indoor spaces. The system uses the Texas Instrument MSP-EXP432E401Y microcontroller, the 28BYJ-48 stepper motor, the ULN2003 driver board and the Pololu VL53L1X time of flight (ToF) sensor.

Table 1: Hardware Components

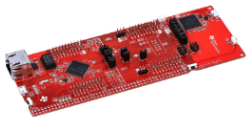
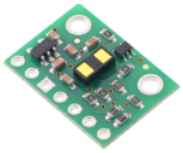
Component Name	**Image**	**Component Datasheet**	**Retail Price**
MSP-EXP432E401Y Microncontroller		https://www.ti.com/lit/gpn/msp432e401y	$62.22
28BYJ-48 Stepper Motor		https://www.digikey.ca/en/htmldatasheets/production/1839399/0/0/1/28byj-48	$15.00 for 5 pcs
ULN2003 Driver Board		https://www.electronicoscaldas.com/datasheet/ULN2003A-PCB.pdf	$11.71 for 5pcs
Pololu VL53L1X Time of Flight (ToF)		https://www.pololu.com/file/0J1506/vl53l1x.pdf	$26.32

Table 2: Required Software

Software Package	**Functionality**	**Link**
Python 3.8	Data Visualization	https://www.python.org/downloads/release/python-380/
Keil uVision5 (MDK)	Microcontroller Program	https://developer.arm.com/Tools%20and%20Software/Keil%20MDK

Features

The performance features of each component of the system are listed below.

MSP-EXP432E401Y Microcontroller

- ARM Cortex®-M4F 32-bit Processor Core CPU (supports floating point operations)
- Accurate structure mapping up to a 4m distance
- Maximum CPU clock frequency 120 MHz
- Bus speed is 24 MHz
- Onboard LEDs provide user visual feedback of start/stop and measurement status
- I2C modules that support high speed mode
- UART communication operating at a 115200bps baud rate
- GPIO pins that support interrupts, PWM, aand ADC

28BYJ-48 Stepper Motor

- Unipolar Stepper Motor
- Full-steps
- Rotates 4-phases completing an 11.25 angle per measurement taken

ULN2003 Driver Board

- 5-12 VDC Supply Voltage
- 4 signal lights onboard

Pololu VL53L1X Time of Flight (ToF)

- I2C Communication Protocol at 100kbps
- Emitter: 940 nm invisible Class 1 VCSEL
- Three distance modes: Short up to 1.3m, Medium up to 3m, Long up to 4m

General Description

This embedded system 3D Spatial Mapping System provides distance measurements and data acquisition essential for creating detailed spatial maps. The project leverages Python 3.8 for data visualization and Keil uVision5 (MDK) for microcontroller programming, ensuring a seamless integration of hardware and software functionalities. The device's compact design, affordability, and efficient performance make it a valuable tool for mapping environments with a focus on practicality.

Some key features of the system are:

- Accurate Mapping: Utilizes a TOF sensor for precise distance measurements, enabling accurate mapping of environments.
- Microcontroller Integration: Seamlessly integrates with a microcontroller for efficient data processing and control.
- Motorized Scanning: Incorporates a motor for automated scanning, enhancing coverage and data collection efficiency.

- Compact Design: Compact and portable design for versatile deployment in various applications.
- User-friendly Interface: Intuitive user interface for easy operation and setup.

Data Flow

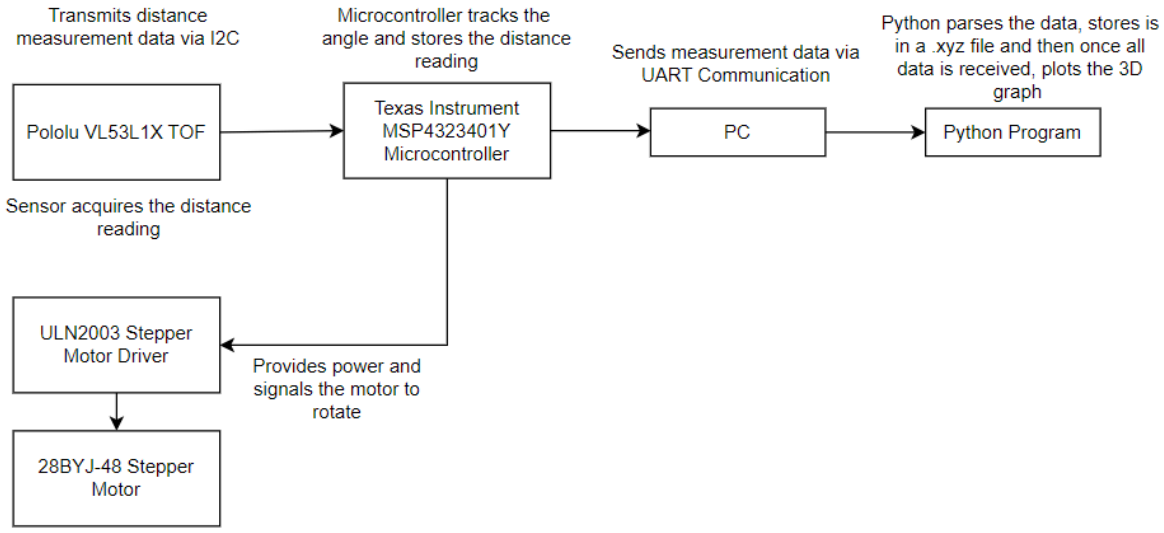


Figure 1: Data Flow Graph

Device Characteristics

Table 3: System Characteristics

Characteristic	**Associated Values**
Time of Flight Sensor Ports	Vin (3.3V), GND, SCL (PortB2), SDA (Port B3)
Stepper Motor Ports	Vin (5V), GND, Port H0:3
Start/Stop Button	Onboard push-button, PJ0
Default Serial Port	COM4
Serial Communication Speed	115200 Baud Rate (bps)
Python Libraries	math, serial, numpy as np, open3d as o3d
Required uC header files	PLL, Systick, tm4c1294ncpdt, vl5311x_api, uart
ToF Range	40 mm to 400 cm

Distance Status Pin	Onboard LED D3

Detailed Description

Data Aquisition

The data used to create the 3D model of the room is acquired via the VL53L1X Time-of-Flight sensor. This sensor chip emits pulses of a 940 nm infrared laser and receives the reflected pulses using a single photon avalanche diode (SPAD) receiver array with an integrated lens. The sensor chip performs multiple functions, including transducing the light signal into a voltage measurement, conditioning the input signal, performing analog-to-digital conversion on the data, and outputting that quantity in millimeters.

Distance Measurement

The time of flight sensor is mounted to the motor shaft with a 3D printed mount allowing it to rotate and take a full 360 degrees scan of the surrounding walls. For every full rotation completed, the motor is programmed to go a full counterclockwise rotation back so the wires are untangled and it can take the next scan from the same start position. The Systick_wait10ms() function is called with an argument of 10 between each step to add a 0.1s delay and allow the motor to stabalize before taking the next measurement. The ToF sensor determines the distance by measuring the time it takes between emitting and receiving an infrared light ray. The calculation for the measurement is:

$$Distance \ = speed\ of\ light\ *\ (time\ of\ travel/2))$$

The program is designed to take 32 distance measurements per full rotation of the stepper motor. The for loop takes 512 steps to do a full rotation and that corresponds to 0.703 degrees per step. To take measurements at every 11.25 degrees, the program utilizes an if statement checking if “i%16 = 0” so that every 11.25 degrees we can call the API function “take_measurement” which signals the ToF sensor to take a measurement.

The data sent from Keil includes RangeStatus and Distance obtained from the TOF's API functions. Each step of the motor increases the loop counter 'i' by one. When 'i' reaches a multiple of 16, the motor has turned 11.25 degrees. Upon completion of a full rotation, corresponding to a 2d layer on the 3d visualization, the program waits for a maximum of 25 seconds (timeout) for another interrupt triggered on PJ0. This allows the user to move the horizontal distance, default is 100cm, before taking the next layer measurements. If the timeout period has passed, the sensor will proceed to send invalid data.

Visualization

The data is visulized in a 2 step-process. First, the microcontroller communicates with the python program via UART and creates a .xyz file based on the data it receives. The default name of this file is

“hallway.xyz” but should be changed for every location you wish to measure so you can retrieve the data and revisit the visualization of each space. Next, it utilizes the Open3D library to create a 3D model of the datapoints.

To determine the XYZ values, the program utilizes trigonometry, specifically computing the sine and cosine of the measurement to get the y and z as shown below. Each layer increments the x value.

```
# Calculate the angle in radians
angle_rad = 2* math.pi *measurement / measurements_per_layer
y = distance * math.cos(angle_rad)
z = distance * math.sin(angle_rad)
x = layer * distance_between_layers  # x-coordinate is the layer index times the distance between layers
```

$$degrees = \frac{360}{32} = 11.25^{\circ},\ step\ size = \frac{512}{32} = 16\ steps$$

Each plane comprises 32 data points, with measurements taken every 11.25 degrees as per system configuration. Open3D is utilized to generate a point cloud from the updated .xyz file. For visualization, a line set is built where each point within a plane is connected to its adjacent point, ensuring continuity. Furthermore, connections are established between planes by linking corresponding points, culminating in the combination of the point cloud and line set for visualization creation.

Application Note

Microcontroller Setup Instructions

Figure 4. shows the circuit schematic for the 3D Spatial Mapping system. Once the hardware connections are in place, you can proceed to the software setup. The softwares used to program the microcontroller and visualize the data are listed in *Table 2.* Start by downloading the required softwares using the links provided. To load the program for the micrcontroller, download the program zip file and open "ProjectDeliverable2.uvprojx" using the Keil uVision5 Application. Once loaded, locate the "ProjectDeliverable_Bonus.c" file using the file navigation bar on the right of the screen. Connect the micrcontroller to the computer using a USB cable and configure the following settings if they are not set already.

Options for Target >> Debug >> Use: "CMSIS-DAP Debugger" >> Settings >> CMSIS-DAP Debugger JTag SW Adapter>> "Any".

Now, load the code onto the controller by clicking each of the buttons shown in *Figure 1* from left to right. Allow time for each process to complete.

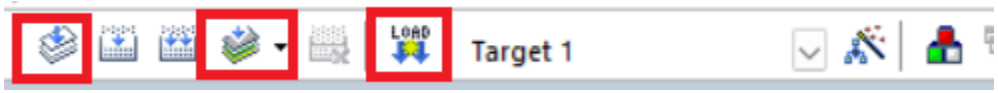


Figure 1: Keil uVision Interface to Load Program

Visualization Program Setup Instructions

To visualize the data, open up "final_3dvisual.py" using Python 3.8 IDE. Press reset on the micrcontoller then configure the correct port for serial communication, eg 'COM4', and set the "total_layers" variable to the number of layers you want to scan (default is 3). Additionally, you can configure "distance_between_layers" which is the horizontal distance you will be moving between each scan (default is 100cm). Save and run the program in python, then when you are ready to scan, click on PJ0 of the microcontroller to start scanning. Once the motor finishes a full rotation and starts rotating back you can move the distance you specified for the next scan. To start the next scan, press PJ0.

Data Collection

The data from the sensor is written to a .xyz file that can be run with the Open3D program to get the 3D model. Ensure that the file name is changed for every 3D scan you would like to do, otherwise it overwrites the file. The default file name is "hallway_final2.xyz".

How to Use

1. Build, upload, and flash the C program onto the microcontroller.
2. Reset the microcontroller by pressing RESET_SW1.
3. Specify the COM port in the Python script and run the script.
4. Initiate measurement acquisition using PJ0.
5. The ToF sensor rotates and pauses every 11.25 degrees to collect measurements, completing a full revolution.
6. Each measurement is displayed in the Python IDLE.
7. Convert each distance measurement from UART serial communication into X-Y-Z Cartesian coordinates.
8. After completing full rotation (1 layer), you can move the distance specified in the file and, press PJ0 to start the next layer, and repeat until reaching the number of layers specified in the file.
9. Once all measurements are gathered, an automated 3D scatter plot is generated. Close the window to view the 3D representation of the data points.
10. To start a new scan, repeat starting from step #2.

Expected Output

A sample of the program output is provided below. For testing purposes, a small hallway on the first floor of ETB was scanned. This hallway had some variations in the ceiling height as well as a curvature leading into one of the classrooms which is effectively captured by the program as shown in *Figure 2* as compared to the location photo, *Figure 3*.

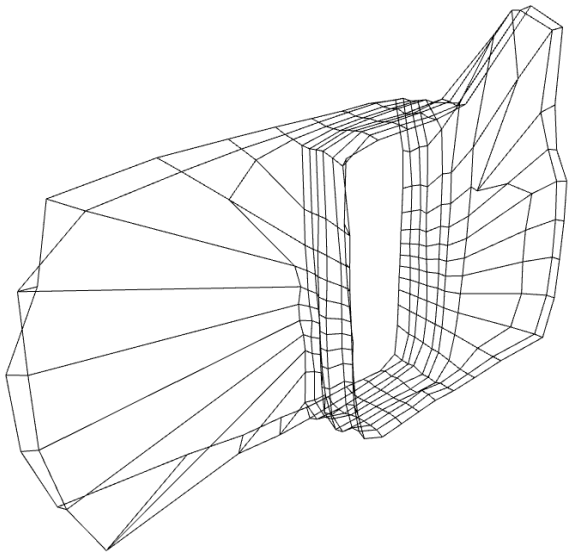


Figure 2: 3D Scan of ETB Hallway - Location E

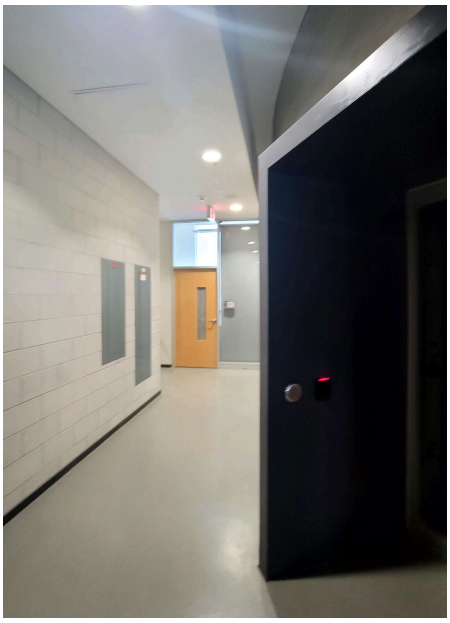


Figure 3: Reference Photo of Location E

Limitations

The MSP432E401Y microcontroller is equipped with a 32-bit Arm Cortex-M4F processor core that has a floating-point unit (FPU), allowing it to perform floating-point calculations within the 32-bit limit. The maximum quantization error for the system can be calculated using the formula:

$$Max\ Quantization\ Error = \frac{Max\ Reading}{2^{\#\ of\ ADC\ bits}}$$

Since the maximum working distance of the ToF is approximately 4 meters, the maximum quantization error can be calculated as:

$$Max\ Quantization\ Error = \frac{4000mm}{2^{16}} = 0.061035$$

The PC's maximum standard serial communication speed is 115200 bps. Any attempt to exceed this rate will result in an error due to the microcontroller's limitation to 115200 bps. This baud rate is crucial for device communication; altering it for one device would disrupt the system's functionality. The microcontroller communicates with the ToF sensor via I2C, using a data line (SDA) for data transmission and a clock line (SCL) for synchronization. The communication speed between these devices matches the ToF sensor's maximum transmission speed of 400 kbps. However, the system's overall speed is constrained by the UART baud rate between the microcontroller and PC, capped at 115200 bps, which is lower than the ToF-to-microcontroller communication speed. Secondly, due to the limited capabilities of the ToF sensor, the device can only map spaces accurately for maximum of 4m in every direction (4000mm).The third limitation is that the stepper motor can experience overheating after extended operation, prompting the user to allow cooling periods between each full run. With regard to the operation of the motor, there is a 5% error in step accuracy that can impact angle precision.

Further, the ToF sensor often struggles to obtain accurate readings when there are, reflective surfaces or glass present, as it misinterprets the light being reflected back , leading to outlier data. External light sources can introduce noise into the sensor's readings, causing unpredictable measurements.

Circuit Schematic

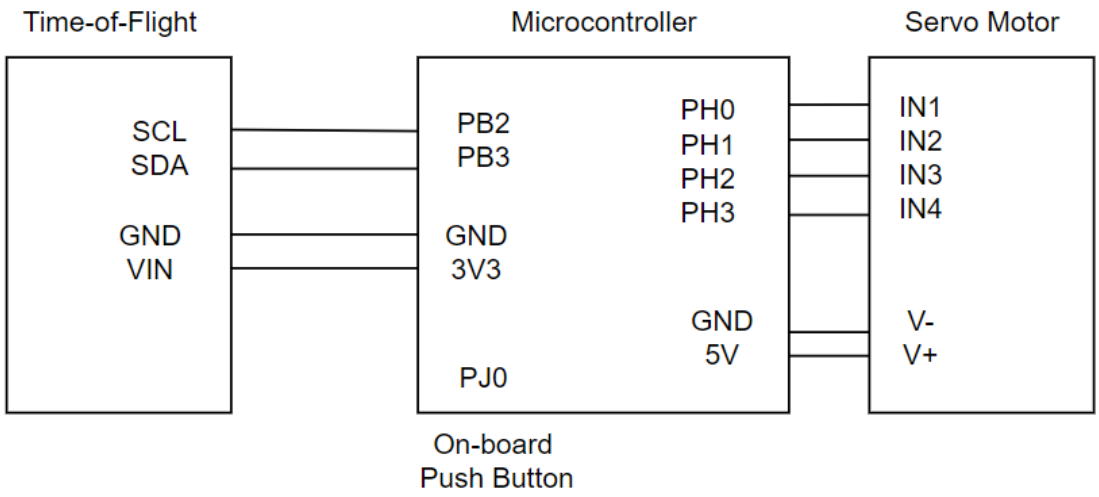


Figure 4: Circuit Schematic

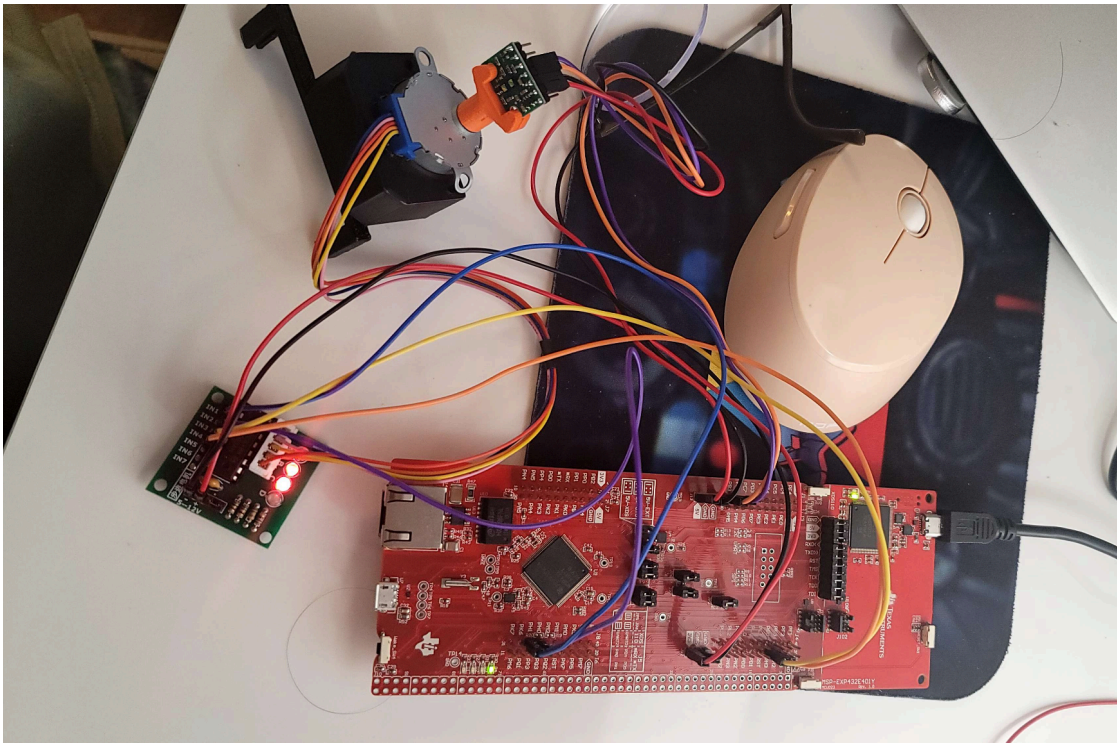
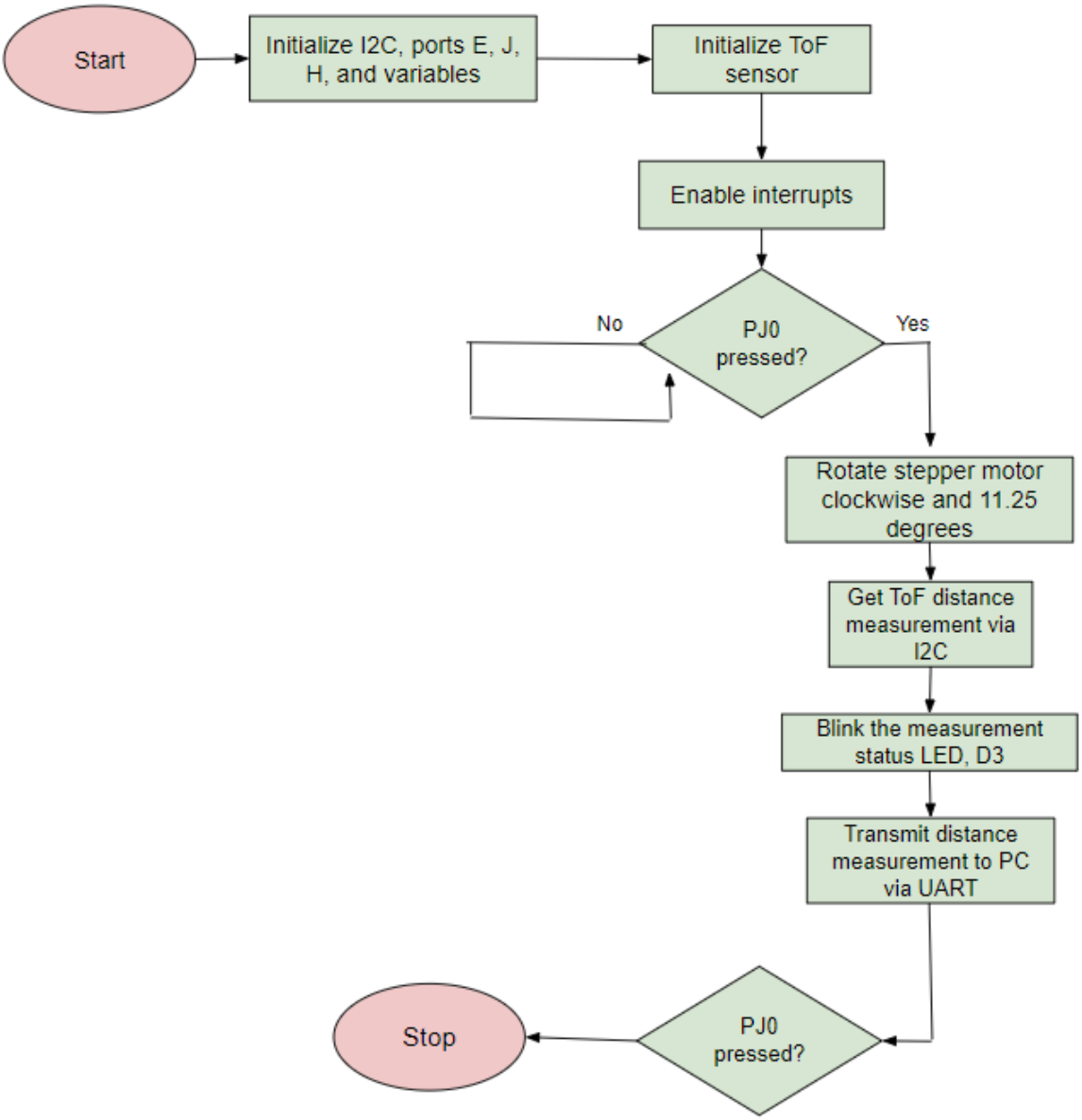


Figure 5: Physical Circuit

Programming Logic Flowcharts

Microcontroller Program



Python Program