1 Device Overview

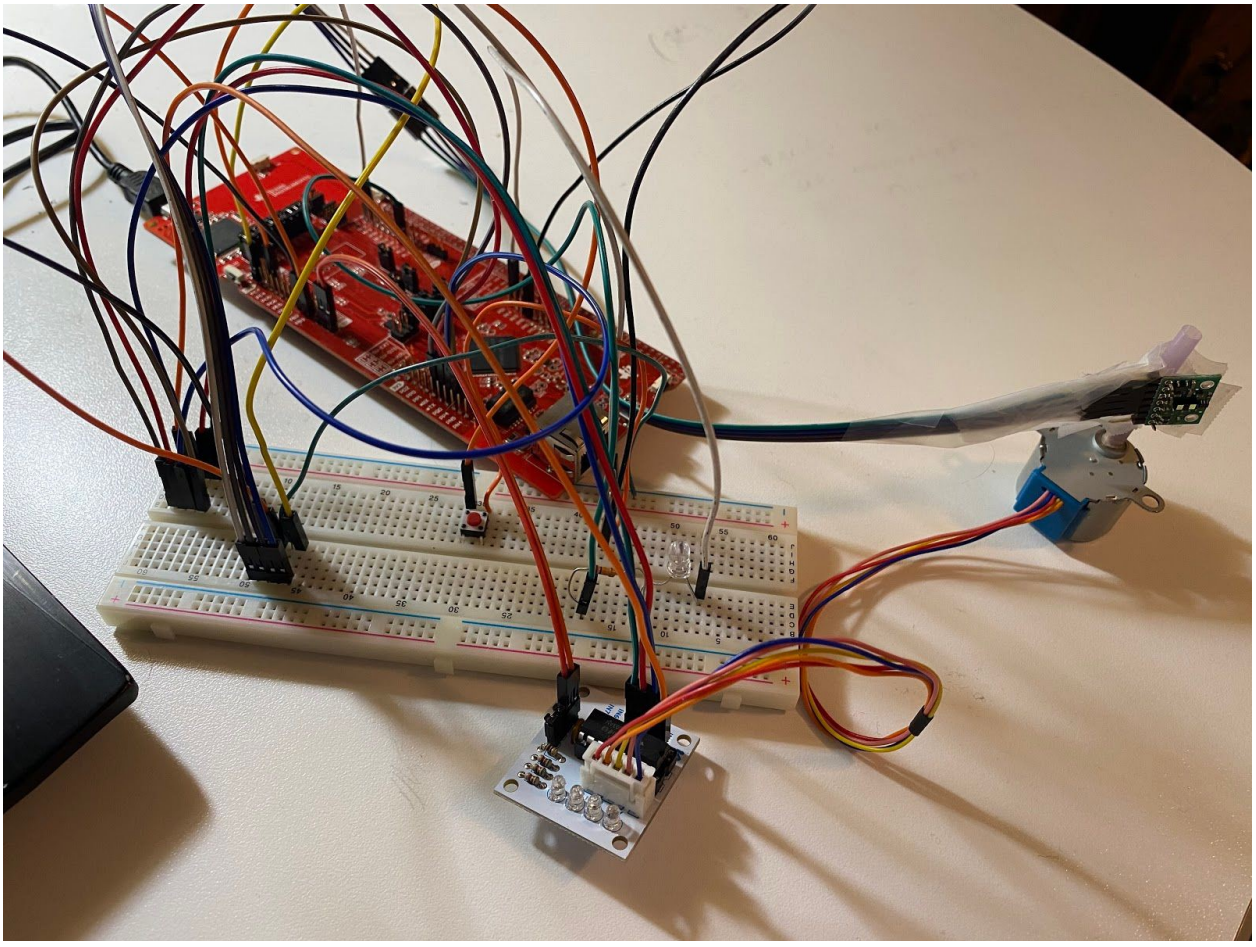

Figure 1: Device

1.1 Features

- Embedded spatial measurement system using time-of-flight to acquire information about the surrounding area
- Replaces the need for Commercial Light Detection and Ranging (LIDAR) equipment, which is expensive and bulky
- Suitable for indoor exploration and navigation
- Employs a Texas Instrument MSP-EXP432E401Y LaunchPad microcontroller
    - The central component; contains the hardware used to transmit data to other systems such as the ToF sensor, stepper motor, and the software used for visualization
    - Default bus speed of 120 MHz
    - Operating voltage of 3.3 V

    - 1024 kB of flash memory, 256 kB of SRAM, and 6kB EEPROM
    - Two 12-Bit SAR-Based ADC Modules, Each Supports Up to 2 Million Samples per Second (2 Msps)
    - Cost is $130
    - Can be communicated with via Assembly Language, however C or any other high-level language can be used, depending on the IDE

- Employs a VL53L1X Time-of-Flight sensor
    - Used to acquire distance measurements
    - Uses laser light (940 nm wavelength) to measure the time it takes to travel between the emitter, to the target, and back to the receiver
    - Up to 400 cm distance measurement
    - Up to 50 Hz ranging frequency
    - Accurate ranging up to 4 m
    - Maximum baud rate of 400 kbits/second

1.2 General Description

This device employs separate components that are connected and communicate with each other. The first component is the VL53L1X Time-of-Flight sensor, which captures distance measurements and communicates these measurements to the microcontroller. The ToF sensor retrieves distance measurements by first acting as a transducer by converting an artificial light signal (energy signal) into an electrical signal. Next, the ToF sensor performs signal conditioning on this analog signal. Finally, the ToF sensor contains an Analog to Digital (ADC) converter, which allows these distance measurements to be converted into discrete values, which ultimately get sent to the user's PC. The Time-of-Flight sensor is mounted on top of a stepper motor, which allows for 360 degree distance measurements. The device is initiated through the use of a push button. Before the button is pressed, an LED on an external breadboard (not on the microcontroller) is lit to indicate that the motor is not rotating, and hence no distance measurements are being taken. After each 45 degree rotation of the stepper motor, an on-board LED is flashed to indicate a distance measurement has been taken. After eight distance measurements have been taken (360 degree rotation of the stepper motor), the motor stops rotating and the off-board LED turns back on. This process will repeat if the button is pushed again. The next component of this device is the MSP-EXP432E401Y microcontroller, which relays information between the Time-of-Flight sensor and the user's PC, via serial communication. Once the ToF sensor outputs digital values, these values can be displayed on a PC using serial transmission. The microcontroller component contains a Cortex M processor, which has 8 UARTs for serial data communication. Serial transmission can occur via simplex communication or half-duplex communication. Ideally, half-duplex communication is used because it allows for more efficient bi-directional communication, due to the synchronization of data from the microcontroller to the PC and vice-versa. Once the PC receives the distance measurements information, a Python program, combined with Open-3D software is used to convert the data into an image.

1.3 Block Diagram

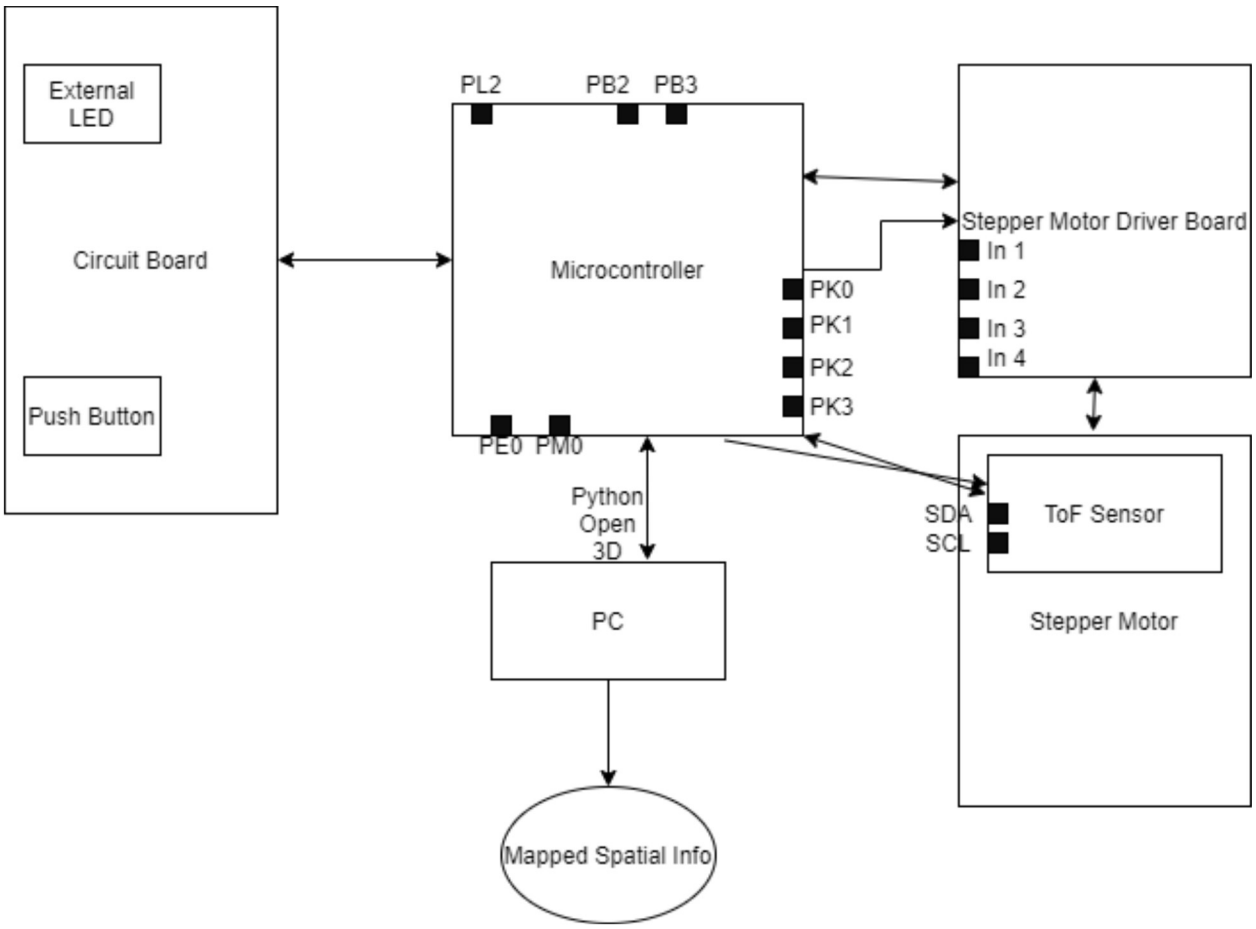

Figure 2: Block Diagram displaying overview of system

2 Device Characteristic Table

This device uses a bus speed of 30 MHz. This can be changed in the code contained in the file named PLL.h. As seen in this code, a bus frequency of 30 MHz corresponds to a PSYSDIV of 15. This can be adjusted according to the commented table at the bottom of the PLL.h file. To get this project up and running, one would need a PC capable of running both the Kiel MDK software development environment, and Python version 2.7, 3.5, or 3.6. The microcontroller pins used, as well as each pin's use in the overall system are shown in Table 1.

Microcontroller Pin	Use
PK0, Pk1, PK2, PK3	Activate coils of stepper motor
PE0	Push-button input
PM0	Push-button output

PB3	Time-of-Flight sensor data signal (SDA)
PB2	Time-of-Flight sensor clock signal (SCL)
PF4	On-board LED
PL2	Powers external LED

Table 1: Microcontroller Pins and Use

The port used for communication with python is COM3 with a communication speed of 115.2 kb/s.

3 Detailed Description

3.1 Distance Measurement

To startup the device, a push-button is used, which has two connections to the microcontroller; one for input into the button and one for output from the button (refer to figure 3). The microcontroller then sends four signals to the stepper motor driver board, which energize the coils of the stepper motor. The Time-of-Flight sensor is mounted to the stepper motor in order to facilitate 360 degree measurements. When the device performs distance measurements, the Time-of-Flight sensor transduces the signal, preconditions the signal, and converts this analog signal to a digital value. The ToF sensor works by calculating distances between the sensor and points on surrounding objects by measuring the round trip time of an artificial laser light signal with a wavelength of 940 nm. This round trip time is the time it takes for the laser light signal to travel from the emitter, to the target, and back to the receiver. The ToF sensor then sends this positioning data to the microcontroller via the SDA line. The formula used to get these distance measurements is as follows:

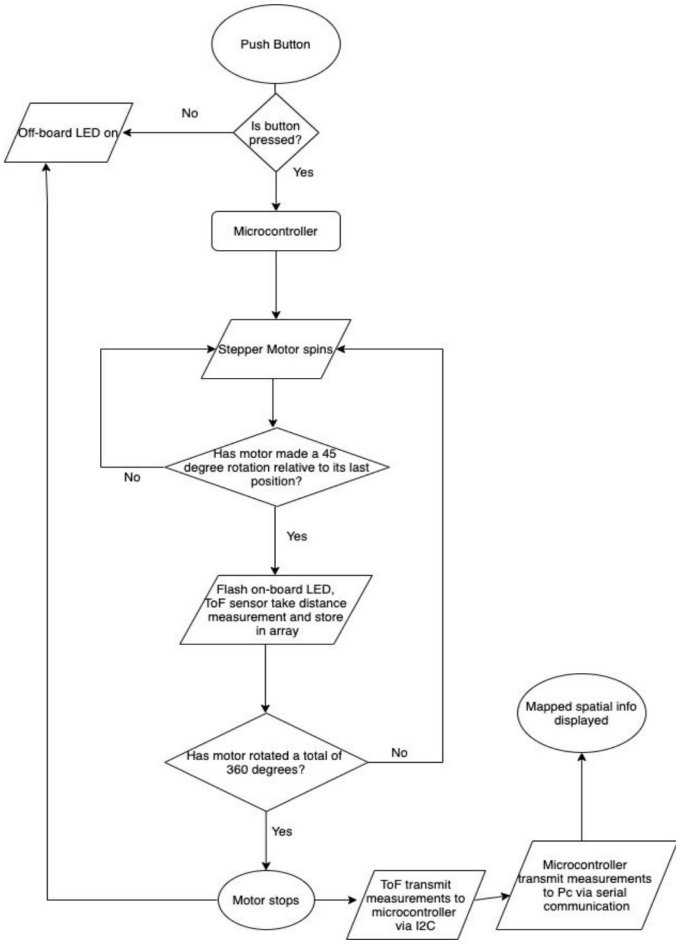

Figure 3: Data Processing

$$d = (c(time))/2$$

where c is the speed of light; c = $3x10^8$ m/s

We divide the distance by two because $time$ is the total roundtrip time, where we only want the time one way (ie from the emitter to the target). After the ToF sensor transduces the laser signal to an analog signal, preconditions this analog signal, and converts this analog signal into digital values, via the ADC converter, this digital data is transmitted to the microcontroller via an I2C bus. As shown in figure 4, data transmitted between the ToF and the microcontroller is synchronized via the SCL line (the clock

signal), and data is actually transmitted, one bit at a time, via the SDA line. Also shown in figure 4, ports PB2 and PB3 on the microcontroller are used to connect with the SCL and SDA lines, respectively, on the ToF sensor. Next, this digital information contained in the microcontroller is communicated to the user's PC, via half-duplex communication. Once this information is passed to the PC, a python program breaks down the distance measurements into y and z coordinates using the following formulas:

$$y = r\cos\theta,\ z = r\sin\theta$$

We now have our coordinates to represent distance (y-z plane), however, we still need the x-coordinate to represent displacement. Since we are not using the IMU sensor to measure displacement, we will increment the x-coordinate by 200 mm each time a y-z plane slice is retrieved. As a result, we now have y-z plane slices for different x-values. Now the python program will work with the open-3D software to create a point cloud, store this point cloud as a file in .xyz format, load the point cloud into an open 3D array, visualize the data as a series of vertices, connect each plane of vertices, connect vertices between planes, and display an image that the user can see. Figure 2 shows how data is processed, beginning from the push-button and ending with an image displaying mapped spatial information.

3.2 Displacement

The use of an IMU sensor to retrieve displacement measurements was neglected in the design of the device. I would have completed my design to measure displacement using the IMU sensor by mounting it onto the stepper motor in a way that it does not rotate with the ToF because it would only be tracking displacement. The IMU sensor would have communicated with the microcontroller in two possible ways; either using the I2C bus at 400 KHz or, more preferably, A serial peripheral interface (SPI) at a frequency of 1 MHz. Much like the ToF sensor, after the IMU sensor has sent data to the microcontroller, the microcontroller would have then sent this data to the user's PC for 3D rendering. Displacement measurements were actually implemented in this project by the ToF sensor taking distance measurements at different x-coordinates. The x-coordinate is incremented by 200 mm each time a new y-z plane slice is acquired. This information will allow us to create a point cloud in python, which can be used to display spatial measurement information.

3.3 Visualization

The computer used to visualize data from this device is a 2018 HP Omen, using the Windows 10 operating system. This computer contains an Intel Core i7-8750H CPU running at 2.20 GHz. This operating system is a 64-bit Operating System. The installed memory (RAM) is 16.0 GB. The program used for visualization is python, and the 3D rendering software used is open 3D. Python versions 2.7, 3.5, or 3.6 could be used to run the open 3D software. The libraries used to visualize the data are the math libraries in python. Data is stored and processed using a .xyz format. Data is visualized using point clouds and vertices, and then connecting the vertices on each plane to produce a 3D image.

4 Application Example with Expected Output

Steps to setup and use device to acquire a signal and map this signal using python:

- Plug in the USB cable, from the microcontroller, into your PC (1)
- Open the kiel project that comes with the device (2)
- Once on kiel, translate and build the project (3)
- Load the project code onto the microcontroller (4)
- Press the push-button on the external breadboard to initiate a set of distance measurements (5)
- Go to the command prompt on your PC and run your version of python (version 2.7, 3.5, or 3.6) (6)
- Choose the COM port setting with the fastest baud rate (7)
- Run the provided python code in order to view an image representing a spatial measurement system surrounding the ToF sensor (8)
- Repeat this process beginning from step 5 to obtain another set of distance measurements for a new spatial measurement system (assuming the device has been moved to a new location) (9)
- With respect to the device's measurements, the y-z plane is used for vertical slices surrounding the ToF sensor, and the x-plane is used for movement forward from the ToF sensor

5 Limitations

1.) Limitations of the microcontroller floating point capability refers to the number of bits that are able to be transferred using UART serial communication. This is because UART serial communication can only transfer 8 bits, however, if the information we want to transfer is greater than 8 bits, the capability is limited. The limitation of the microcontroller in regards to its use of trigonometric functions is that the answers are not exact; values are rounded. This is because only radians are supported in the imported math library, therefore the value of pi needs to be used. However, this value needs to be inputted manually resulting in a coordinate calculation that is not the exact same as what one would get if the actual value of pi was available.

2.) Max quantization error = resolution (Δ) = $V_{FS} / 2^m$

For the ToF module, $\Delta = 3.5 / 2^8 = 0.014$
It should be noted that the operating voltage ranges from 2.6 V - 3.5 V and the I2C can be 8, 16, or 32 bit. Therefore, to maximize the quantization error we will use 3.5 V for V_{FS} to maximize the numerator and 8-bits to minimize the denominator.

For the IMU module, $\Delta = 3.6 / 2^{16} = 5.5x10^{-5}$

3.) The maximum standard baud rate that can be implemented to communicate with the PC is 115.2kb/s. This was verified through the use of RealTerm.

4.) The communication method used between the ToF sensor and the microcontroller was an I2C bus, via the SDA line. The communication speed is 400 kbits/s

5.) The element which is the primary limitation on the speed of the entire system is the stepper motor. The reason for this is the time it takes for the coils to energize in the stepper motor to cause it to rotate is much longer than the time it takes for the microcontroller to transmit the data to the stepper motor or the ToF to acquire a distance measurement. This limits the whole system, as distance measurements could be captured in a shorter period of time if the system was not limited by the delay of the stepper motor.

6.) Based upon the Nyquist rate, the necessary sample rate required for the displacement module would vary based on how many times it is programmed to capture a displacement measurement. If, for example, the IMU is programmed to capture 20 measurements every second, then the necessary sample rate can be described by the following:

$$f_{\text{sample}} \geq 2 * f_{\text{signal}}$$

$$\mathbf{10\ Hz} \geq f_{\mathbf{signal}}$$

6 Circuit Schematic

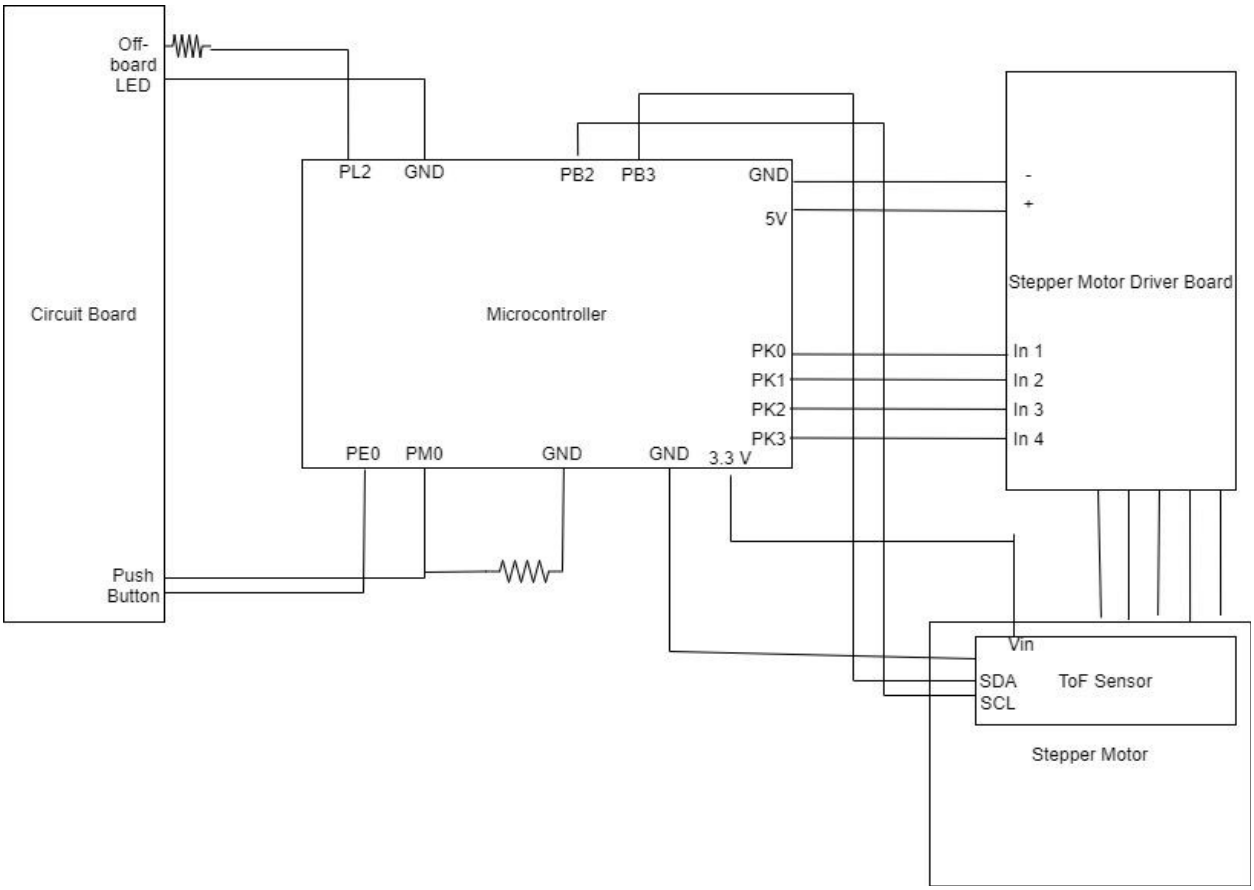

Figure 4: Circuit Schematic