JARZYNOWSKI SEMICONDUCTOR

D2 – Measurement LED
D3 – Status LED
60Mhz – Bus Speed

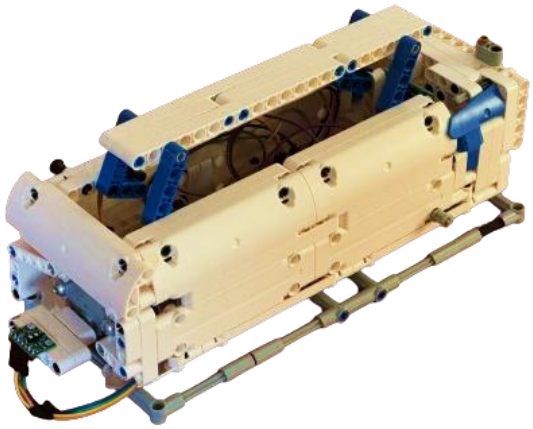
2023/2024 2DX3 Final Project Specification
April 2024 / Revision (B – Deliverable 2)
Matthew Jarzynowski / jarzynom / 400455803

[JSCS] – JARZYNOWSKI SPATIAL CAPTURE SYSTEM

Integrated 3D-Spatial Mapping System Using Time-Of-Flight Tech.

View Online

Feature Overview



- Low-power microcontroller with highly sophisticated 3D-visualization firmware to map measurements within near real-time, relative.
- Integrated Texas Instruments (TI) 32-bit ARM Cortex-M4 based MSP432E401Y microcontroller set at 120 Mhz with 1024 KB flash, 256 KB SRAM [1].
- Mounted VL53L1X Time-of-Flight (ToF) sensor using FlightSense™ ranging technology, operating with a range of 4 metres (m) at a clocked ranging frequency relative to 50 Hz [2].
- Rotary mechanism driven by a programmed stepper motor and driver board with a custom angle offset.
- Onboard normally closed (n.c.) push-buttons (Reset, PJ0, PJ1) programmed using an interrupt-based approach to achieve functions of device reset, motor start/stop, and plot collected spatial positions.
- Keil uVision 5 integrated development environment (IDE) used to program C-based firmware.
- Analog Discovery 2 (AD2) and relevant Waveforms software [4] used to analyze bus and signal frequency.
- Communication protocols of Inter-Integrated Circuit (I2C) and Universal Asynchronous Receiver and Transmission (UART) used to send and receive data packets at 115200 bits-per-second (bps).
- MATLAB-based plotting script to input transmitted spatial measurements over serial UART connection.

General Description

J-SCS, referred to by its full name as the *Jarzynowski Spatial Capture System*, is a fully end-to-end integrated 3D mapping solution. The system works by using a VL53L1X Time-of-Flight (ToF) module in combination with a DC stepper motor-driven rotary shaft to take relative distance measurements every ~3 degrees [6].

These measurements are communicated via the I2C protocol to the integrated MSP432E401Y microcontroller, from Texas Instruments. To communicate the data to a computer, UART is used over a serial connection through a USB cable connected to the back of the microcontroller. Custom MATLAB code interprets these distance values, returned in millimetres (mm) and converts them into relative cartesian coordinates, based on each layer offset, X-value. These coordinates are then plotted within 3D space, with surfaces calculated between each layer [6].

Since J-SCS uses simple components as well and the enclosure itself is made of LEGO™ *Technic* pieces, creating a strong outer "exo-structure," it can be argued that such a system is better than commercial alternatives. This simpleness allows for many of these scanners to be used in tandem, say for example when mapping out the entire floor of a building such as the *Information Technology Building* (ITB) [6].

Data-Flow Block Diagram

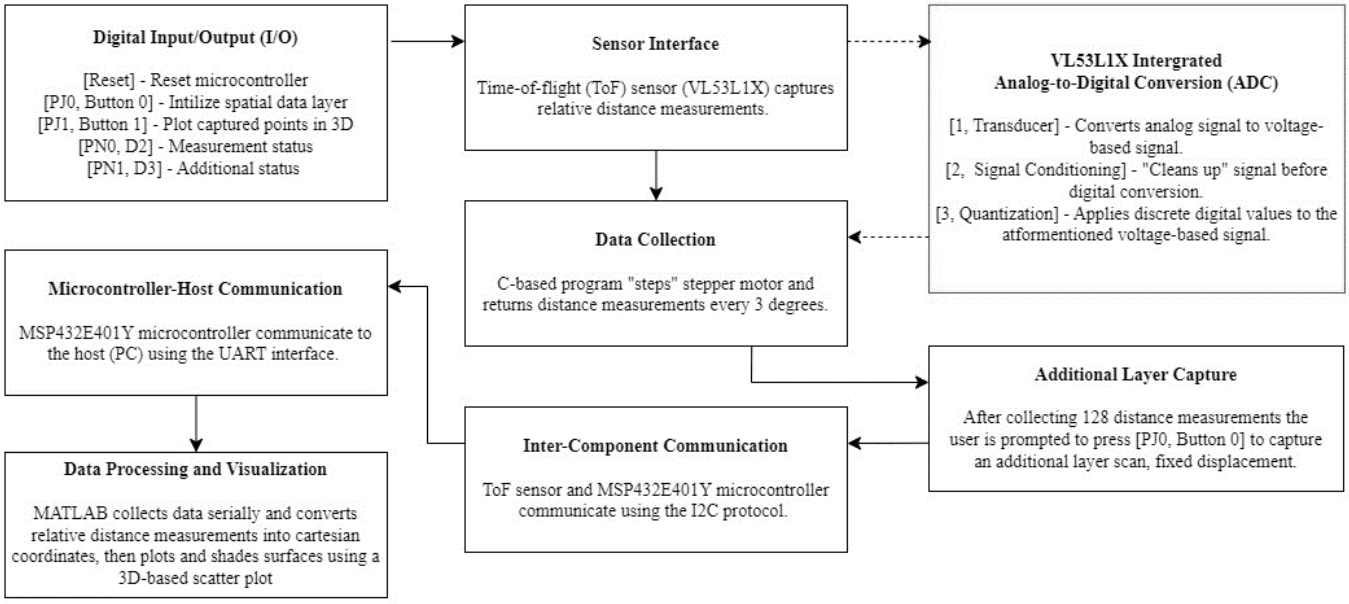


Figure 1. The dataflow process quantified based on each relevant step.

Device Characteristics Table

Characteristic	Unit of Measure	Specified Value
Dimensions	mm x mm x mm (LWH)	270 x 95 x 97 (± 1 mm)
Weight	g (grams)	~472 (± 1 gram)
Bus Speed	MHz (mega-hertz)	60
Digital I/O Status LED	-	PN0 – LED (D2)
Digital I/O Additional LED	-	PN1 – LED (D3)
Microcontroller Pins	-	PM(0-3), PB2, PB3, 3.3V and GND (Ground)
Baud Rate	bps (bits-per-second)	115200
Microcontroller [1]		
Manufacturer	-	Texas Instruments (TI)
Model Number	-	MSP432E401Y SimpleLink™
Processor Family	-	ARM Cortex-M4
Communication Protocols	-	I2C, UART
Base Clock	MHz (mega-hertz)	120
Operating Voltage	V (volts)	3.3
Temperature Range	°C	-40 to 105
Memory	KB	1024 (Flash), 256 (SRAM)
ToF Sensor [2]		
Manufacturer	-	STMicroelectronics
Model Number	-	VL53L1X FlightSense™
Emitter Type	nm (nanometer)	940, Class 1 infrared-based laser
Ranging Frequency	Hz (hertz)	50
Maximum Ranging Distance	cm	400
Stepper Motor [3]		
Model	-	28BYJ-48
Phases	-	4

Distance Measurement – Radial Data Collection

To collect relative distance measurements, the VL53L1X time-of-flight (ToF) sensor emits an infrared-based laser pulse, in the Class 1 laser range of 940 nm [2]. These pulses are captured by the integrated single photon avalanche diode (SPAD) receiver array, which accepts the incoming signal and takes note of its exact timing [2]. This timing is then multiplied by half of the total time taken, to give a relative "time of flight," and through the use of a look-up table (LUT), an exact milli-metre (mm) distance can be outputted. It can be noted that the maximum range for the ToF is 400 cm (4 metres) before distance falloff as well as that the ranging frequency is 50 Hz, rather than the traditional 60 Hz for such sensors. The sensor itself communicates with the MSP432E401Y microcontroller through the use of the I2C protocol.

Distance Measurement - Data Processing and Coordinate Conversion

Data processing begins with collecting the relative distance data. This process starts with resetting the microcontroller using the rear-facing push button, SW1, or switch 1, normally closed (n.c.). After this is done the ToF sensor is initialized and is ready for distance measurement capture. Using UART, the ToF sensor displays its relevant status serially to communicate to the user when its default setup is successful. Once a clear status message has been sent, the microcontroller waits for an input on Button 1, PJ0, left relative. This button, as described earlier, in *Figure 1*, begins the relative layer capture for the specific displacement.

Once an interrupt is detected on PJ0, Button 1, the additional status LED, D3, is flashed to acknowledge the interrupt, and the stepper motor begins to rotate, or "step" every 3 degrees. With each step, the motor stops for approximately 0.2 s (seconds) and waits for the ToF sensor to take an accurate distance measurement. The motor waits until the measurement LED, D2, is flashed before continuing onto the next step. This process is repeated for 128 measurements, or however many are specified at the top of the custom-defined C-script. These measurements are communicated between the ToF sensor and microcontroller using the I2C protocol, and displayed serially, within real-time using UART onto a serial terminal.

In terms of coordinate conversion and relative displacement, since these distance measurements are radial, simple trigonometry can be employed to calculate the missing Y and Z coordinates, since X is known as the displacement. These can be calculated by the following formulae;

$$(1)\ \mathrm{Y} = distance \times cos(\theta)$$
$$(2)\ \mathrm{Z} = distance \times sin(\theta)$$

These values are returned also through the serial terminal, through the use of the UART protocol. Regarding displacement, the first 128 measurements taken by the ToF sensor have a relative displacement of 0, meaning that X itself is set to 0 mm. After each 128-measurement layer is taken, X is incremented by 100, simulating a step taken, which is approximately 10 cm. This stream of data, X, Y and Z coordinates is again, sent using the UART protocol to the serial terminal, to be further processed by MATLAB for 3D plotting.

It can be noted that this data processing and conversion is happening on the MSP432E401Y microcontroller itself, rather than the personal computer (PC), where the final outputted UART transmission only contains the coordinates themselves, rather than the steps in between. An example of a sample transmission for an X displacement of 300 could be;

$$300, -76.412, 86.913$$

Where each coordinate follows the syntax of X, Y, then Z, and where X, the displacement, is incremented every 128 measurements, or every scanned layer. Note that the distance is rounded to 3 decimal places, for general convenience in plotting within MATLAB.

Data Visualization

To plot the relevant data in near real-time, a MATLAB script is employed. This script is run after ToF sensor initialization and runs in the background collecting each data point communicated over UART, serially. It can be noted that no additional software is necessary for this visualization, as everything is handled within MATLAB itself.

The script begins with receiving the relative coordinates over a specified communication port, COM port, serially, where it is then parsed, in that the script runs through UART output and separates each coordinate stack. These stacks are saved into a text file, named "data.txt," where such a file can be thought of as a database for coordinates. After scanning the specified number of layers, the script reclusively calls another plotting script that uses the aforementioned collected data, from "data.txt" and plots it using a 3D-based scatter plot. To achieve a look similar to that of the hallway, surfaces are calculated between each tri-intersection of data points. It can be noted that the script itself does not know the displacements, and X-values, beforehand, but rather uses the outputted UART transmission as a basis for its X-axis, since the X-value is incremented every 128 measurements. It can also be said that the MATLAB script itself does not have any error handling logic, while the C-based program does to account for windows and or other anomalies.

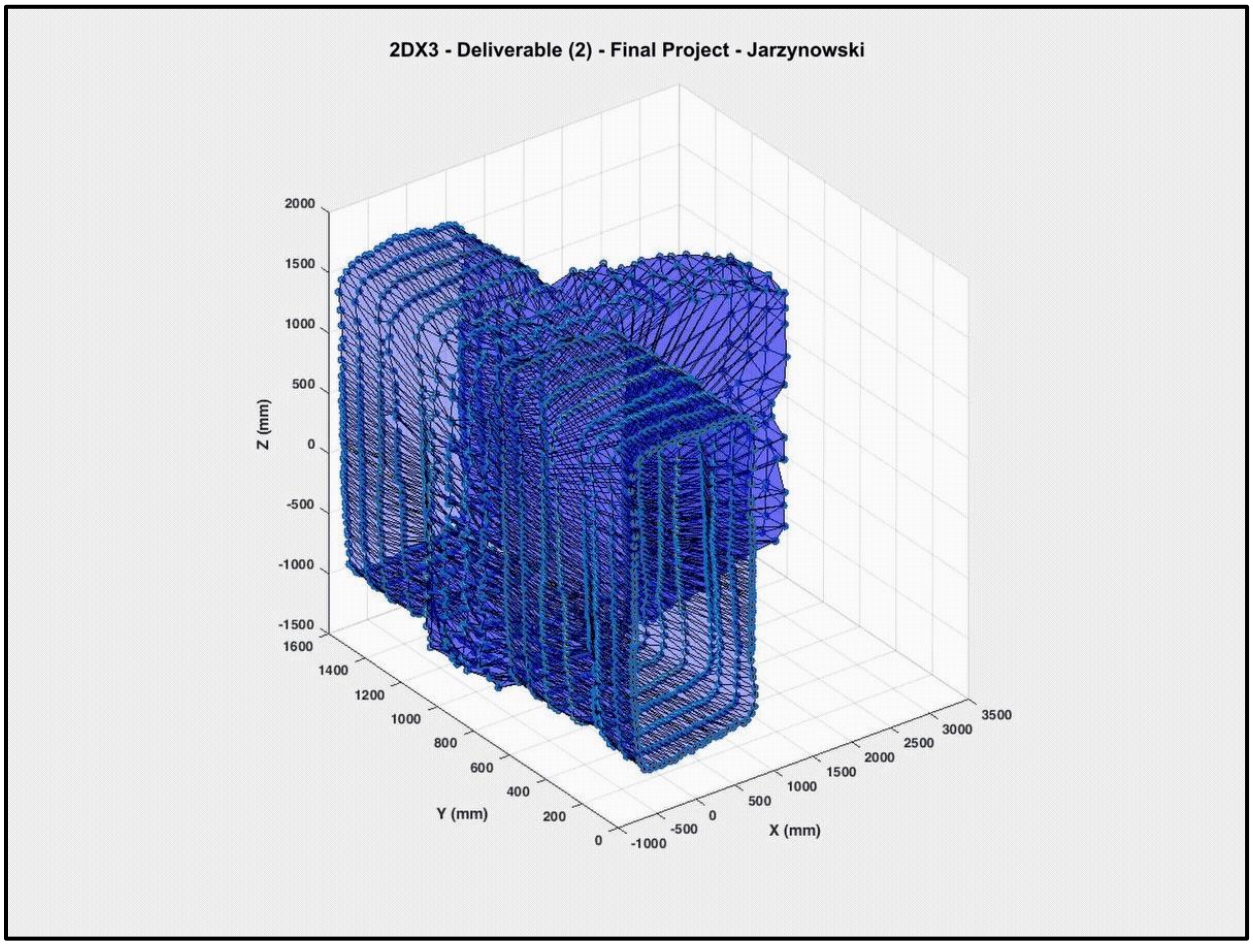


Figure 2. 3D Representation of Hall (D), within ITB

Application – Instructions and Expected Output

Consider the following a simplified "instruction set" on how to use J-SCS;

1. Build, load, and flash the C-based script using Keil uVision 5 onto the MSP432E401Y microcontroller.
2. Reset the microcontroller by pressing the rear-facing reset button, SW1.
3. Determine the COM port used, and change its relevant values in the MATLAB script, line 9.
4. Run the MATLAB script after successful sensor initialization.
5. Press PJ0, Button 1, to begin scanning layer 0, displacement of 0. Observe the output within MATLAB's built-in terminal, and confirm using intuition.
6. After 128 measurements are taken, confirm that the data has been collected by opening "data.txt" within the root folder of the MATLAB script.
7. After the first layer is scanned, press PJ0, Button 1, to begin scanning the subsequent layer, and so forth until the entire area has been scanned.
8. Once no input has been detected for approximately 40 seconds, the MATLAB data collection script will terminate and recursively call the plotting script.
9. Depending on processor clock frequency, and speed, the expected output should be a 3D spatial visualization of the scanned area, that will open in an external window, adjacent to that of the running script.
10. The program can be reset by resetting the microcontroller itself and beginning from Step 4.

Brief Users Guide

1. **Getting Familiar with J-SCS**
   Begin with getting to know how J-SCS works, its control systems (buttons), and its underlying spatial data capture system. Read through the provided literature found in the Data-Flow Block Diagram, *Figure 1*, and the relevant methods for distance measurement and visualization, pages 3 and 4.

2. **Initial Setup and Assembly**
   For proper function of the J-SCS system, the wiring and overall assembly of the device are critical. Consult *Figure 5* and *Figure 6* for correct wiring and overall assembly "buildout."
   - Connect a micro-USB (Type-B) cable from the back of the device to a USB (Type-A) port to a personal computer (PC).
   - As the assembly is pre-built, consider checking all connections following the circuit schematic as seen in *Figure 5*.
   - As a final check, ensure that the pins of the stepper motor are connected following that in *Figure 6*, as well as that it is using the 5 V pin on the microcontroller.

3. **Software Installation and Configuration**
   Ensure that a relatively new version of MATLAB is installed, R2024a as of April 2024, and that Keil uVision 5 is installed, from ARM. Also, to find the relevant COM port within *device manager*, follow the path, only applicable to Windows-based machines.

   $$Start \rightarrow Device\ Manager \rightarrow Ports\ (COM\ \&\ LPT) \rightarrow XD2110\ Class\ Application/User\ UART$$

   Note down the COM port number and refer to Step 3 of the instructions above to change it within the MATLAB script.

4. **Confirmation**
   While each distance measurement is being taken, confirm using intuition that the values being received within the MATLAB terminal make sense, and can be confirmed visually. Before scanning a large area, place your hand in front of the sensor and take a scan. You should be able to see your hand within the plot, in that there will be a section that is close relative to the surrounding area.

General Troubleshooting Guide

Consider this a general "toolkit" to help solve common issues with the J-SCS system;

1. *Doesn't turn on*: Ensure proper wiring and connectivity by rewiring and referring to *Figure 5*.
2. *Microcontroller turns on but motor doesn't rotate*: Make sure that the motor is connected to 5 V.
3. *Wire keeps getting tangled*: Move the ToF mount to start North relative, facing the ceiling.
4. *Push button PJ0, Button 1, doesn't work*: Reset the microcontroller and restart data collection.
5. *Sensor doesn't initialize*: First check pin wiring, then replace connecting wires (internal breakage)
6. *Stepper motor board has no lights*: Ensure that the +/- leads are connected, common as they are loose.

ITB - Hall (D)

Figure 3. Photo of the assigned hallway for capture, Hall (D) in ITB.

Spatial Visualization

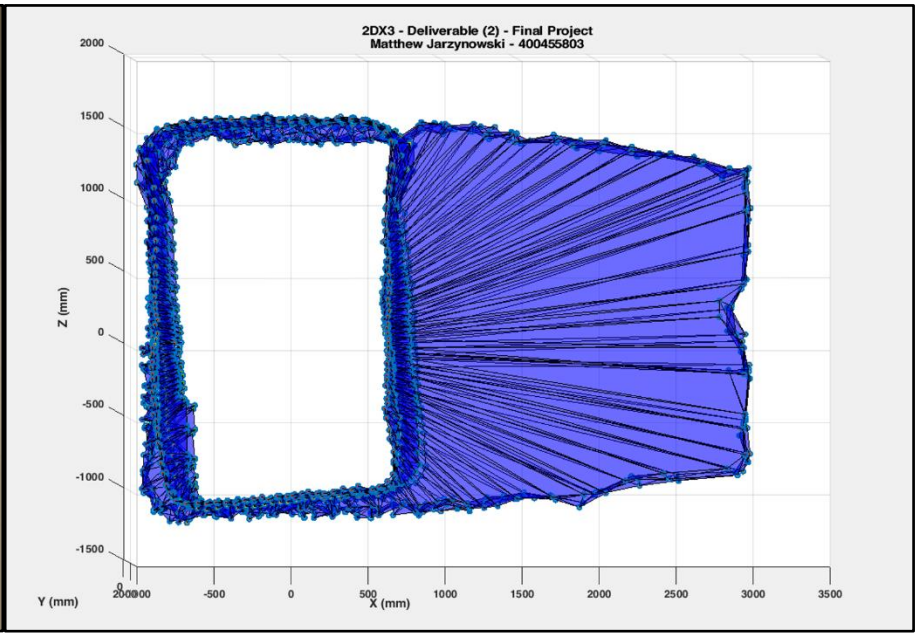


Figure 4. 3D MATLAB plot of the assigned hallway, Hall (D) in ITB.

Project Limitations

1. **Limitations of the microcontroller floating point capability and use of trigonometric functions.**
   It can be noted that the MSP432E401Y microcontroller does have a built-in floating-point unit (FPU) which augments its capabilities to handle single-precision floating-point operations [1]. It can also be said that, on such a low-power device as the microcontroller, trigonometric operations are usually handled through the use of a look-up table, LUT, which increases overall performance as the computations are precalculated and derived from memory. In terms of calculating the Y and Z values, from the relative distance, R, it can be said that there will be some limitations in terms of overall performance, but since the flashed program is of C-nature, C allows for the use of floating-point data types, there is no real lose of accuracy. It can be said that every radix point should be considered, but in this application accuracy to the nearest millimetre is "good enough" to represent a scan of a metre-wide hallway.

2. **Maximum quantization error of each of the Time-of-Flight (ToF) modules.**
   The relative quantization error for the VL53L1X ToF module is based on environmental lighting conditions and the reflectivity of the surface being measured. It can be said that accounting for all lighting conditions, the error is within the range of ± 20 – 25 (mm), or can be computed by the formula;

   $$Ranging\ Error\ =\ Accuracy\ +\ Repeatability\ Error$$

   As found in the VL53L1X datasheet, provided by STMicroelectronics [2].

3. **What is the maximum standard serial communication rate you can implement with the PC? What speed did you implement and how did you verify?**
   The maximum standard serial communication rate can be found by connecting the microcontroller to the personal computer (PC) and viewing its baud rate (bps) through *device manager*. Consider the path of Start → Device Manager → Ports (COM & LPT) → XD2110 Class Application/User UART, then finding the connected COM port, COM # → Properties → Port Settings, whereby opening the dropdown menu it can be seen that 128000 bps is the maximum serial communication speed.

4. **What were the communication method(s) and speed used between the microcontroller and the ToF modules?**
   The I2C protocol is used to communicate between the microcontroller and the ToF module at a transmission speed of 100 kbps (kilobits per second).

5. **For the entire system, which element is the primary limitation on system speed? How did you test this?**
   The primary limitation on system speed is the set PSYSDIV clock, as changing it from the default clock of 120 MHz, PSYSDIV 3, to the assigned frequency of 60 MHz, PSYSDIV 7, should double the period. This was confirmed through the use of an AD2 oscilloscope where a port was set to change between binary 0 and 1 every clock cycle. This was confirmed as when the frequency decreased by 50%, henceforth the period increased by 50%, relatively speaking.

6. **Show the steps and calculations to configure your assigned system bus speed.**
   My student number is 400455803, where position J-relative is value 3, thus referring to Table 1 in the project specification, the assigned bus speed is 60 MHz. To configure this, within Keil, one would navigate to *PLL.c*, then *PLL.h*, and change the relevant PSYSDIV variable to 7, as defined by the comments in lines 49 – 59 respectively. Within *SysTick.c*, the value in line 70 in *SysTickWait* would change to 60 MHz.

Circuit Schematic

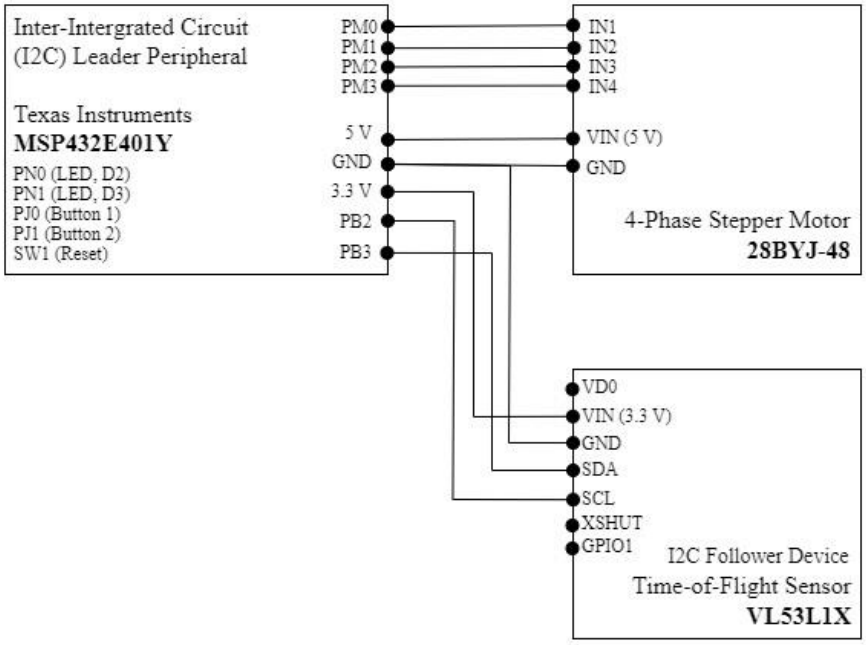


Figure 5. Block diagram for the J-SCS System, in terms of its components.

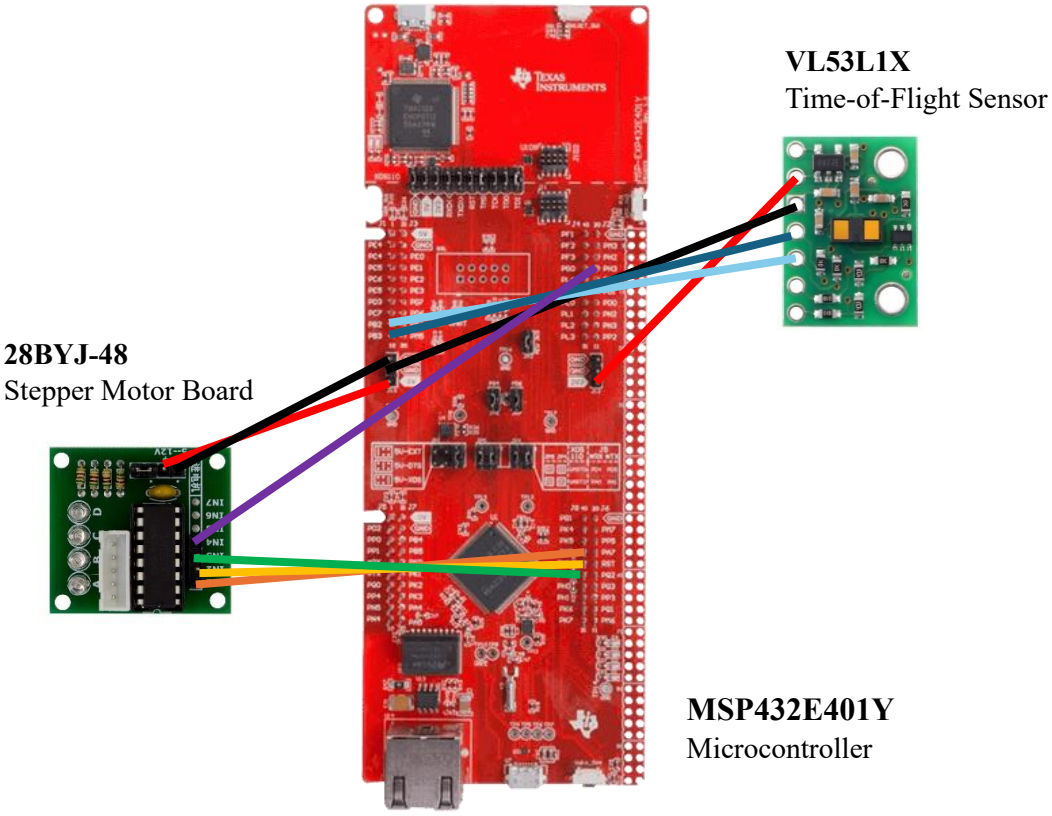


Figure 6. Circuit schematic of the J-SCS System, in terms of its real-world wiring.

Programming Logic Flowcharts

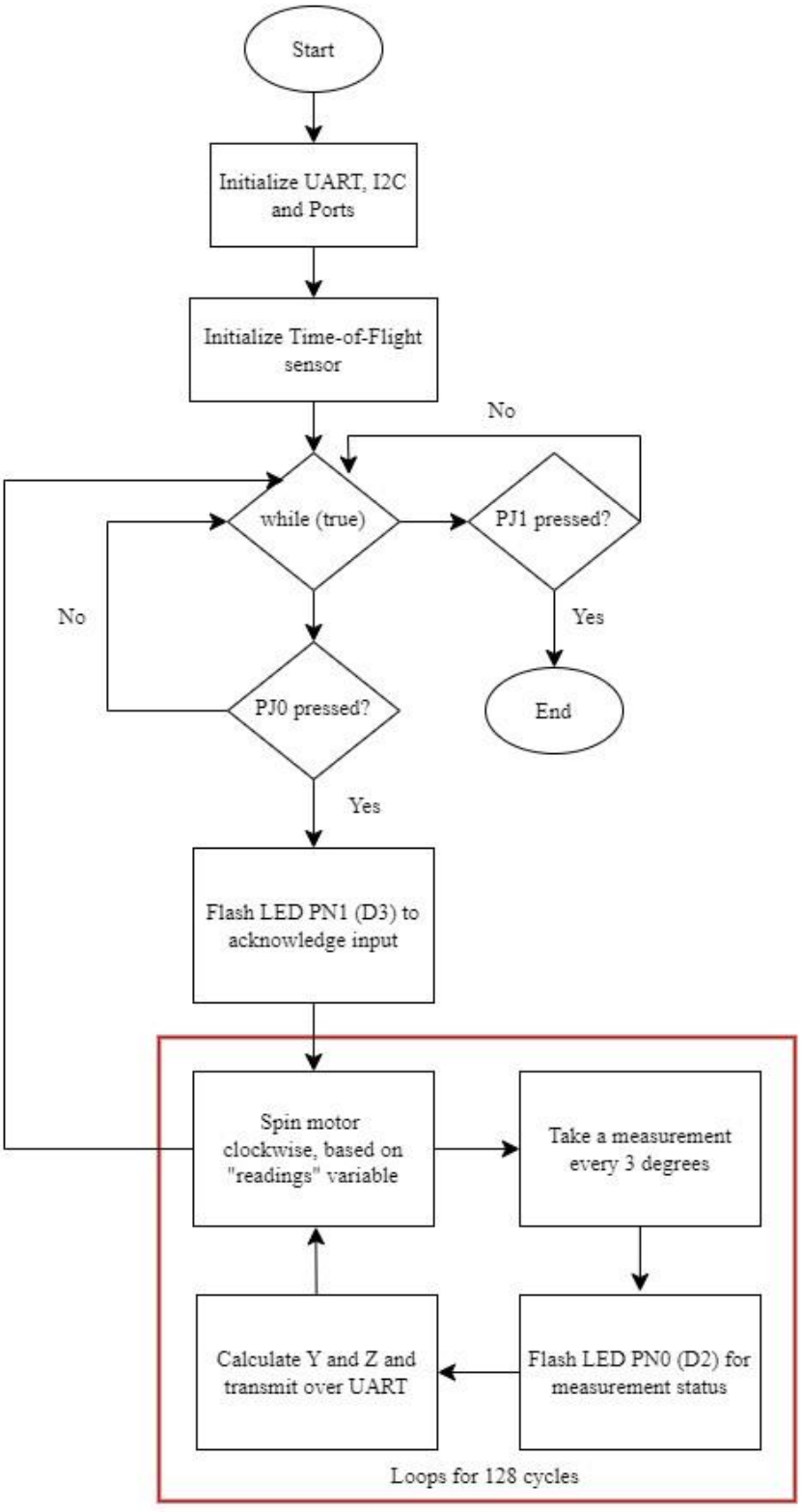


Figure 7. Flowchart for C-based script, running on the microcontroller.

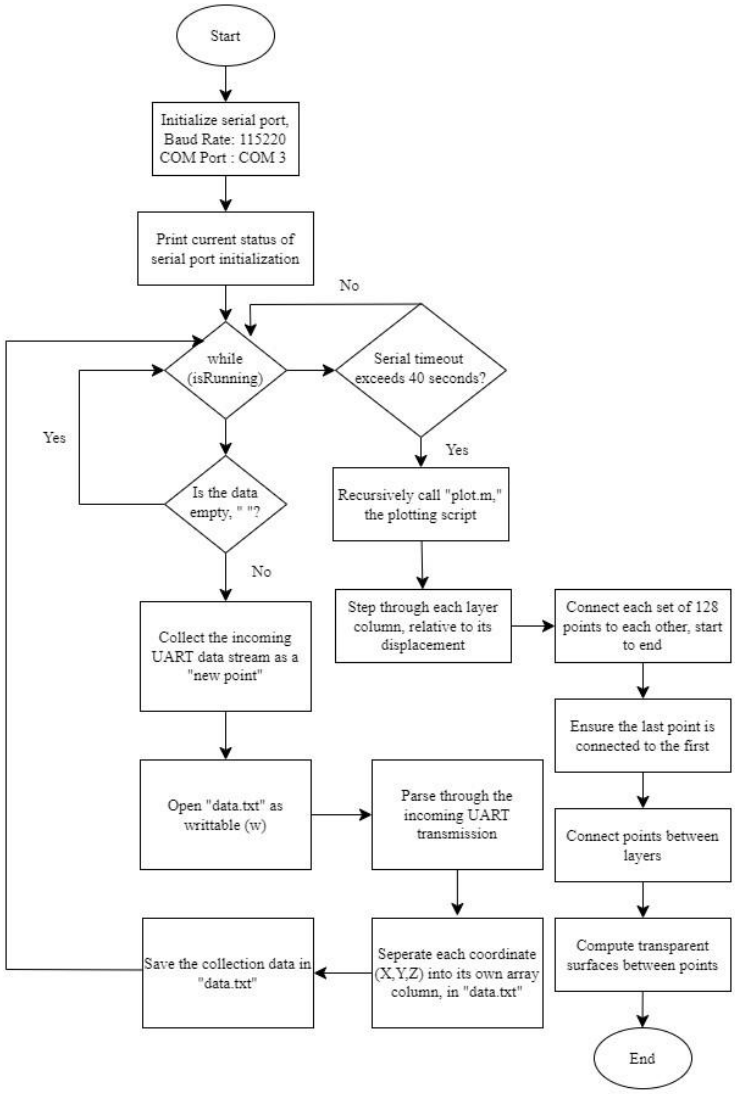


Figure 8. Flowchart for MATLAB script, responsible for spatial visualization.

References

[1] “MSP432E401Y SimpleLink Development Microcontroller,” Texas Instruments (TI), https://www.ti.com/product/MSP432E401Y (accessed Apr. 17, 2024).

[2] “VL53L1X Datasheet,” STMicroelectronics, https://www.st.com/en/imaging-and-photonics-solutions/vl53l1x.html (accessed Apr. 17, 2024).

[3] “28BYJ-48 Stepper Motor Datasheet,” Components101, https://components101.com/motors/28byj-48-stepper-motor (accessed Apr. 17, 2024).

[4] “Analog Discovery 2 (AD2),” Digilent, https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/ (accessed Apr. 17, 2024).

[5] “COMPENG 2DX3:Microprocessor Systems Project Avenue.” https://avenue.cllmcmaster.ca/d2l/le/content/557632/Home (accessed Apr. 17, 2024).

[6] “Jarzynowski.com,” Matthew Jarzynowski, https://www.notion.so/mjarzy/J-SCS-Jarzynowski-Spatial-Capture-System-b0514a11aff147b4b596ef5d07e7d35f (accessed Apr. 17, 2024).