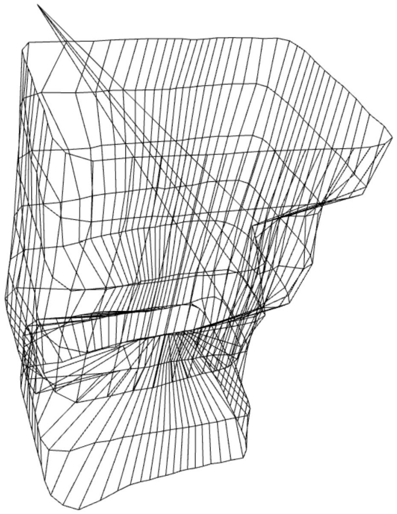
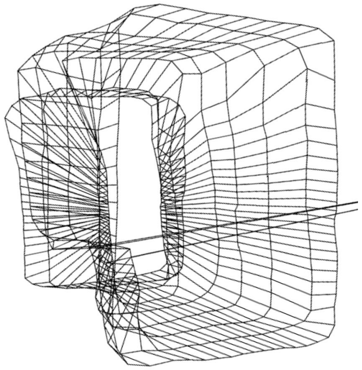
3D Spatial Mapper: An Embedded System Design

Project by: Veronica Marrocco





Project by:
Veronica Marrocco

Date:
04/25/2025

Table of contents

01 Background

02 System Overview

03 Data Acquisition

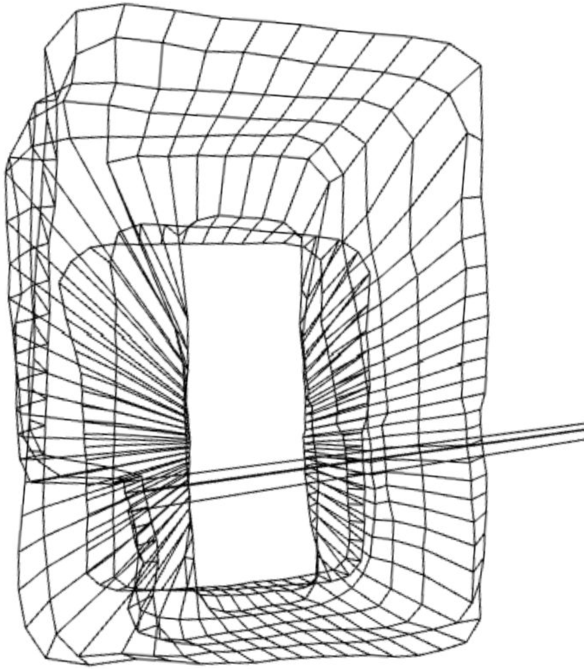
04 Data Visualization

05 Challenges

06 Reflection

01

Background



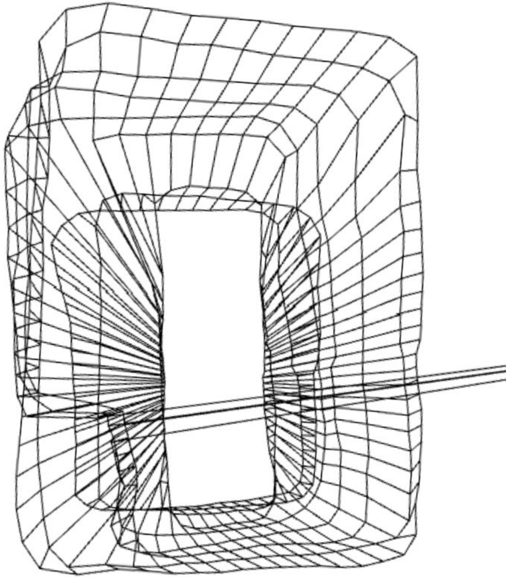
Design Challenge & Constraints

Goal: Design and build a realtime, embedded spatial measurement system using the MSP432 microcontroller, a time-of-flight sensor, and a rotary mechanism to provide 360 degrees of visualization within a single vertical geometric plane.

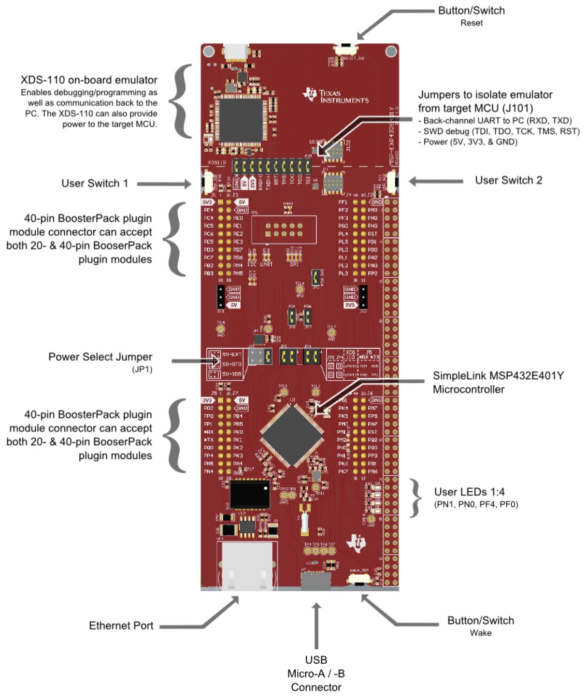
Power supply	Powered via microcontroller USB (3.3V and 5V rails)
Microcontroller Limitations	• 32-bit ARM Cortex-M4F • Only UART/I2C for communication • 60 MHz Bus Speed (Assigned, Configurable with PLL Registers)
Budget	Total cost within $150
ADC	No onboard ADC or DSP - ToF sensor must do its own digitizing
Resolution Targets	• Scan depth of 3 meters • <15deg angular resolution (>24 scans per revolution)

Hallway Scan

Test Plan: Each student was assigned a particular hallway in the University to capture in 3D.



TI MSP-EXP432E401Y LaunchPad

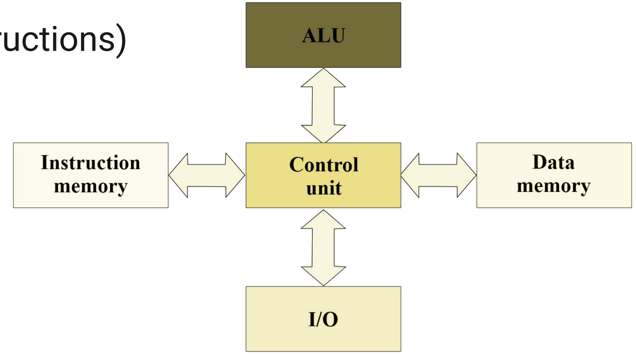


- 3.3V or 5V supply
- 120MHz maximum bus speed, derived from **PIOSC** (16MHz) or **MOSC** (25MHz) by configuring PLL registers
- Several Peripherals (e.g., ADC, PWM, GPIO)
  - **14-bit SAR ADC**
  - 128 I/O pins, organized into **8-bit GPIO ports**
  - **24-bit down counter** (SysTick Timer) to create specific delays & generate periodic interrupts

    Write delay value in reload register…

    $DELAY = (RELOAD+1) \times T_{SYSCLK}$
- Single precision floating point unit (**FPU**)
- Vector interrupt table (**NVIC**):
  - Decides where processing should start on boot up
  - Maps interrupt sources to ISR addresses & their priorities (max 111 interrupts)

MSP432 Memory Organization



- Harvard memory architecture (separate buses for data & instructions)
- **32-bit words** and **32-bit addresses**
- Little Endian (LSB in lowest address byte)
- Memory divided into blocks (flash ROM, SRAM and MPU)
- **MPU** (memory management unit) contains **registers**
  - Start at 0x400.0000
  - r0-r12: general purpose
  - r13: stack pointer alias of banked registers
  - r14: link register holds address of current instruction for easy return access following interrupt
  - r15: program counter keeps address of next instruction to be executed
- Registers are also divided into groups
  - **register address = base address + offset**
  - Importantly, RCGCGPIO reg with offset 0x608 from system control registers enables & disables GPIO ports

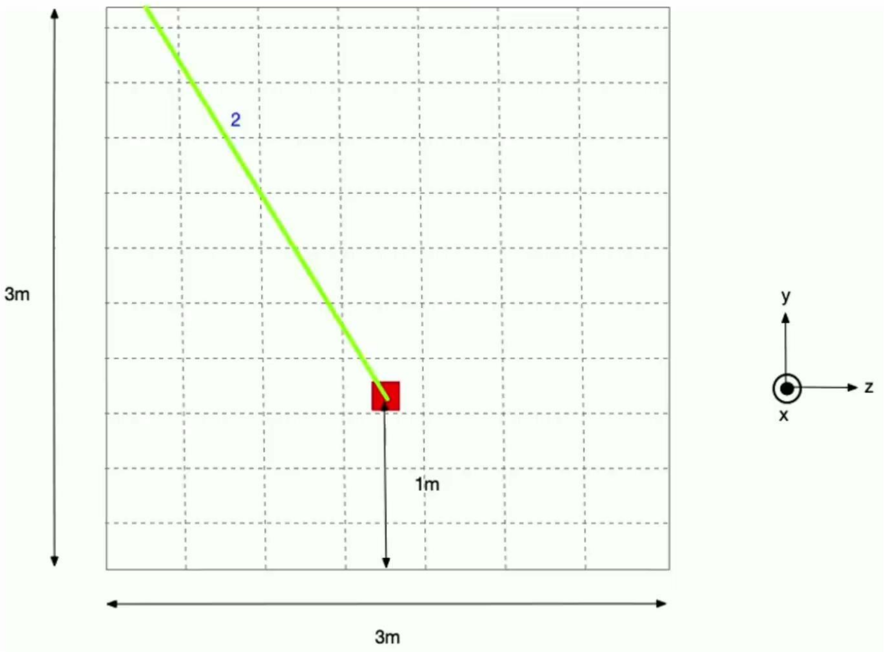
Mapping Strategy

- Move along x-axis manually, scan in y-z planes
  - Avoids need for expensive linear stage hardware
  - Easy to coordinate
- Coordinate system:

$$\mathbf{x} = n \cdot \text{increment}$$

$$\mathbf{y} = d \cdot \sin(\text{theta})$$

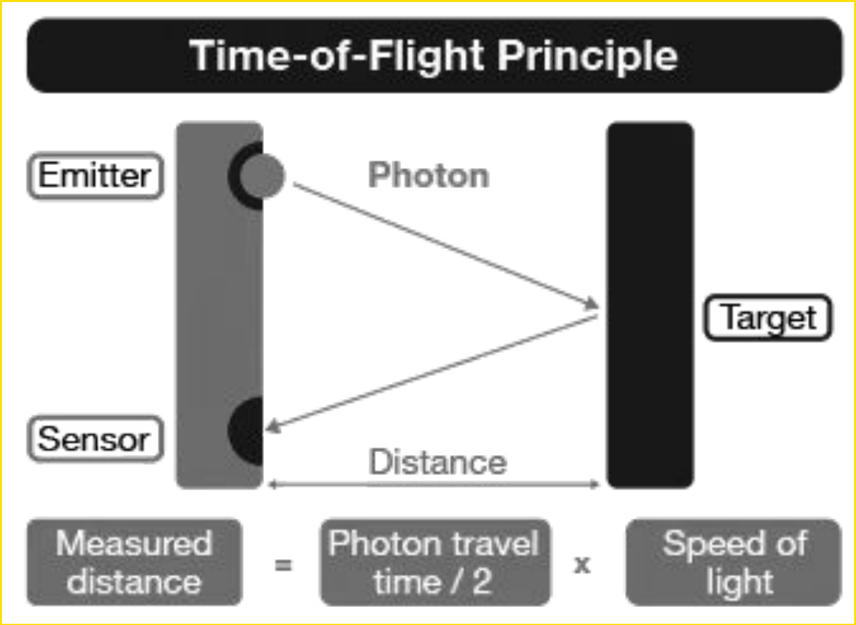
$$\mathbf{z} = d \cdot \cos(\text{theta})$$



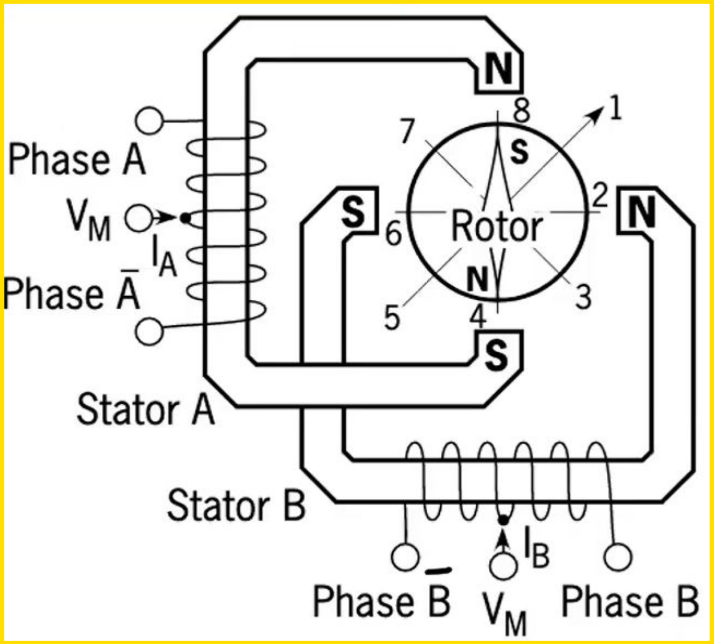
Time of Flight Sensor

- *A ToF sensor* emits IR light and measures its return travel time to compute object distances.

Adafruit's VL53L1X



Stepper Motor



Full Step CW Sequence:
A → B → A' → B'

- Ideal where **spatial precision** matters more than speed/torque
- Operation is **inherently digital** (control the motor by energizing coils in a specific sequence using GPIO)
  - "Energize" = logic low
- Setup: Permanent magnet rotor surrounded by stator coils

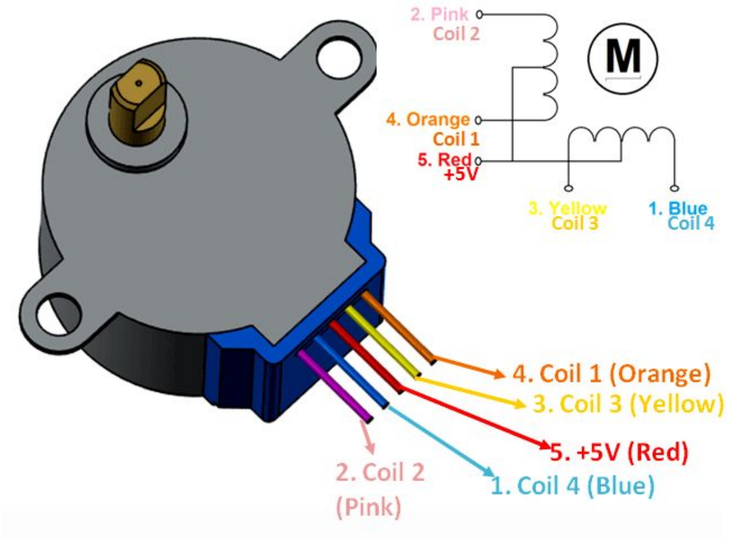
I/O to A, A', B, B'...

- ⇨ Control direction of current flow
- ⇨ Control polarity of stator electromagnets
- ⇨ "Pull" rotor along discrete steps

Part Selection: 28BYJ-48 Stepper Motor with **ULN2003** driver board.

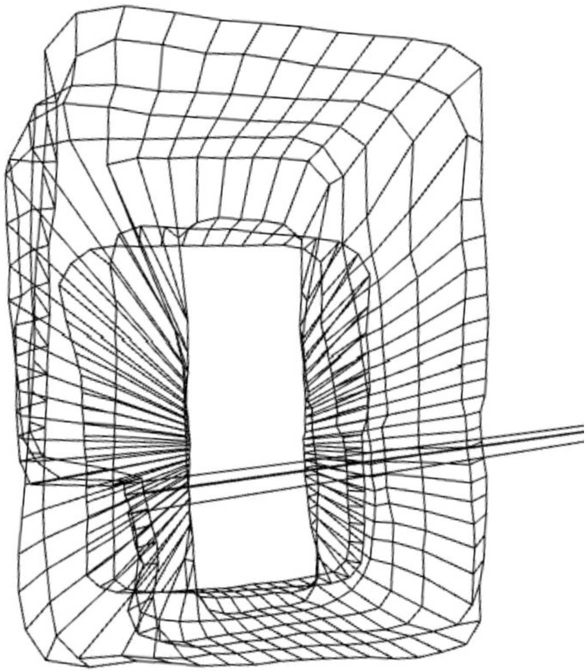
Stepper Motor

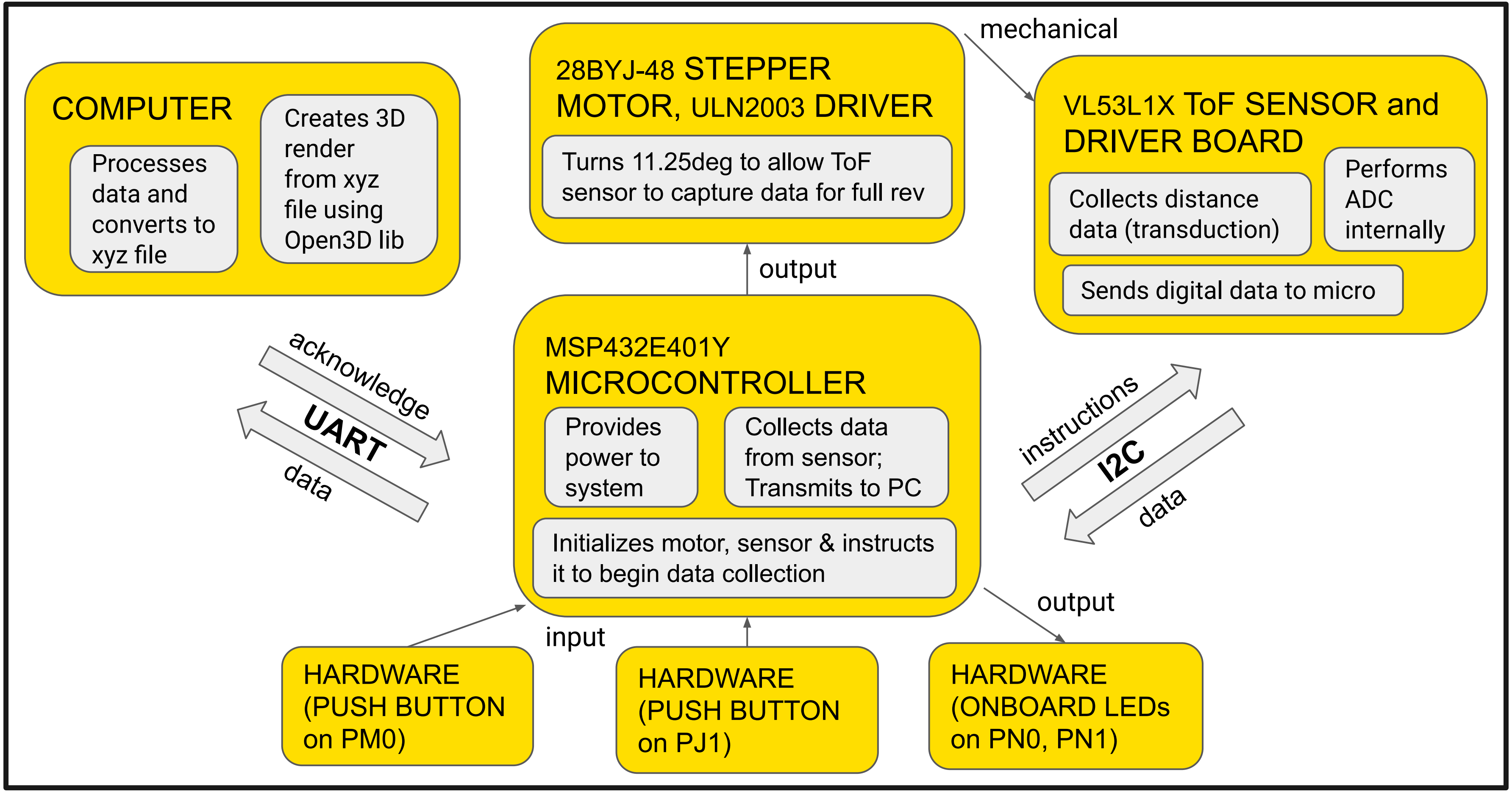
- Unipolar (only ever energize one side of each coil at a time)
- Internal gear ratio of 64:1
- Use full-step control: energize pairs of coils in a 4-phase sequence to achieve rotation of 11.25 deg increments
- Each 11.25 step takes 64 internal full steps due to gear ratio
- CW rotation sequence is A → B → A’ → B’



02

System Overview

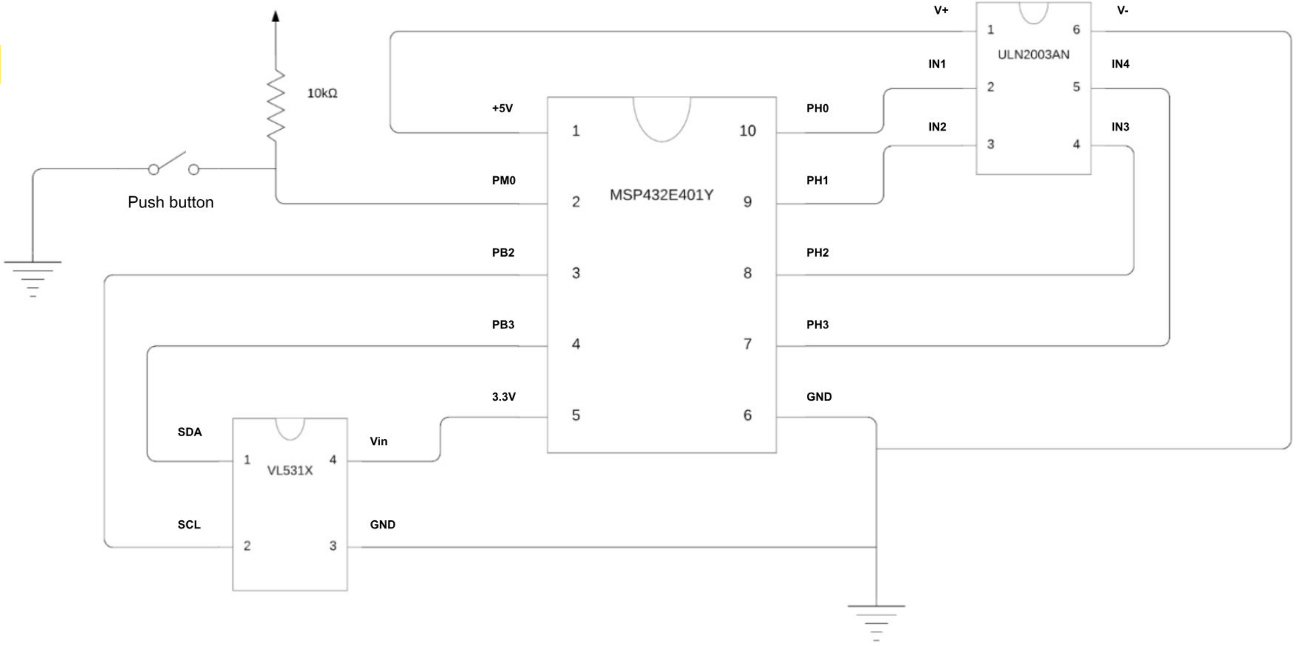



COMPUTER
Processes data and converts to xyz file
Creates 3D render from xyz file using Open3D lib
28BYJ-48 STEPPER MOTOR, ULN2003 DRIVER
Turns 11.25deg to allow ToF sensor to capture data for full rev
mechanical
VL53L1X ToF SENSOR and DRIVER BOARD
Collects distance data (transduction)
Performs ADC internally
Sends digital data to micro
output
acknowledge
UART
data
MSP432E401Y MICROCONTROLLER
Provides power to system
Collects data from sensor; Transmits to PC
Initializes motor, sensor & instructs it to begin data collection
instructions
I2C
data
input
output
HARDWARE (PUSH BUTTON on PM0)
HARDWARE (PUSH BUTTON on PJ1)
HARDWARE (ONBOARD LEDs on PN0, PN1)

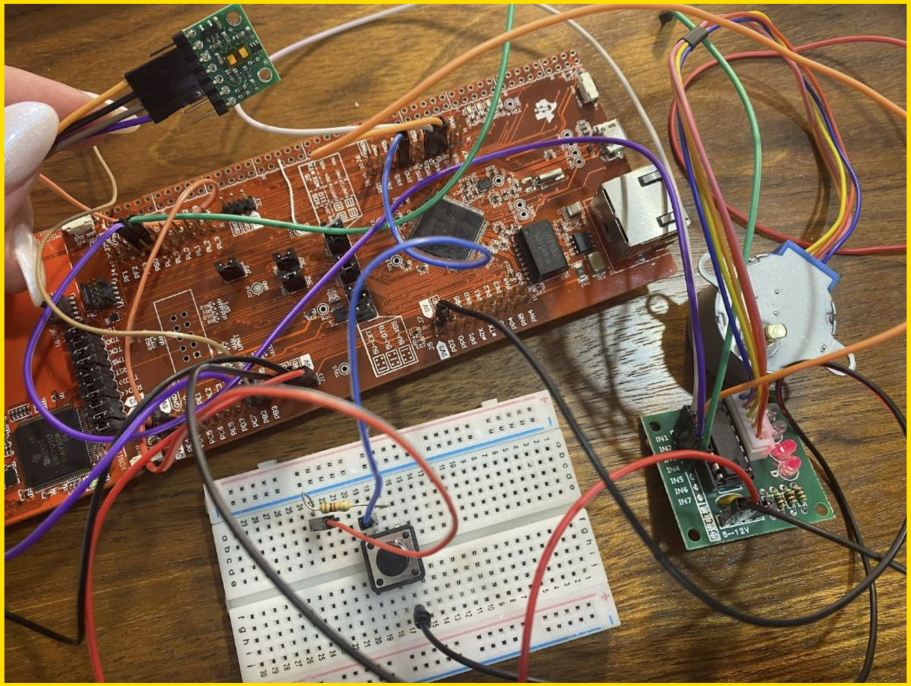
Circuit Schematic

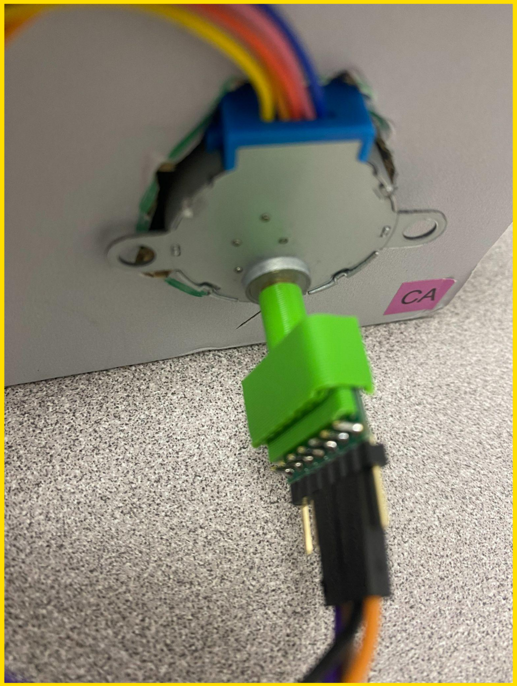
Onboard elements (not pictured):

- **Pull-Up resistors** for I2C lines on PB2, PB3
- **Status LEDs** on PN1 (for on state) and PN0 (to flash each time 11.25deg step complete and sensor is about to take measurement), and PF0/PF4 for successful boot msg from sensor
- **Push button** on PJ1 for interrupt-driven pause/resume functionality



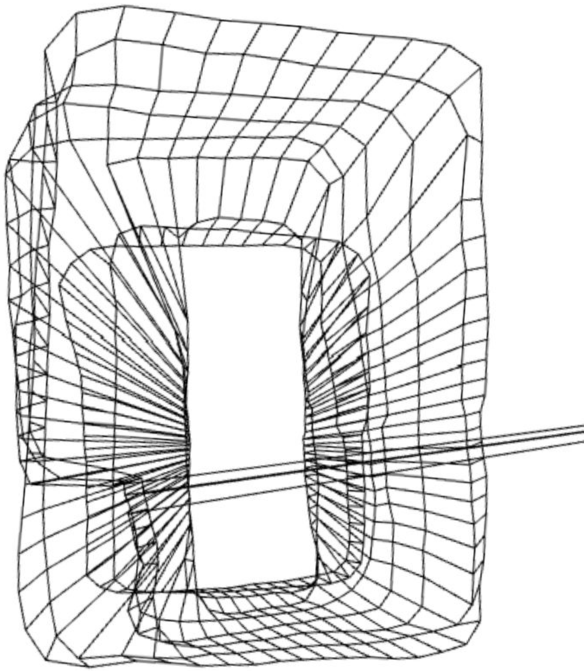
3D Mount & Hardware Setup





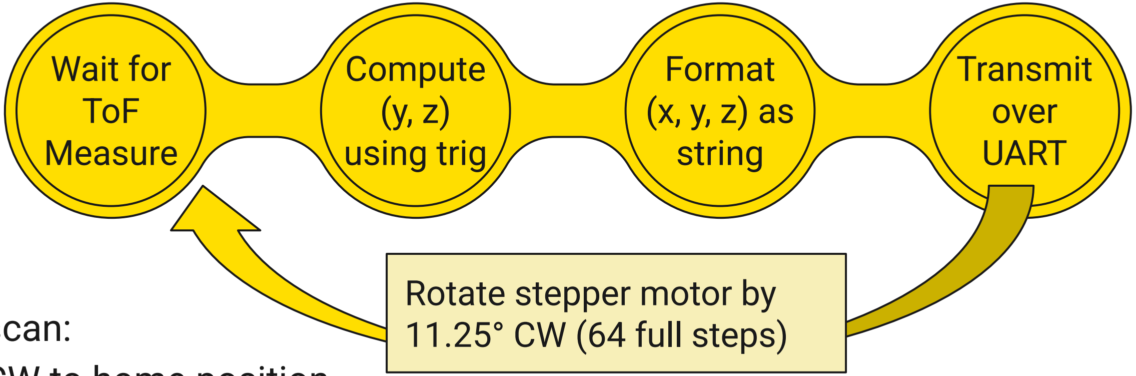
03

Data Acquisition



Firmware Program (in C)

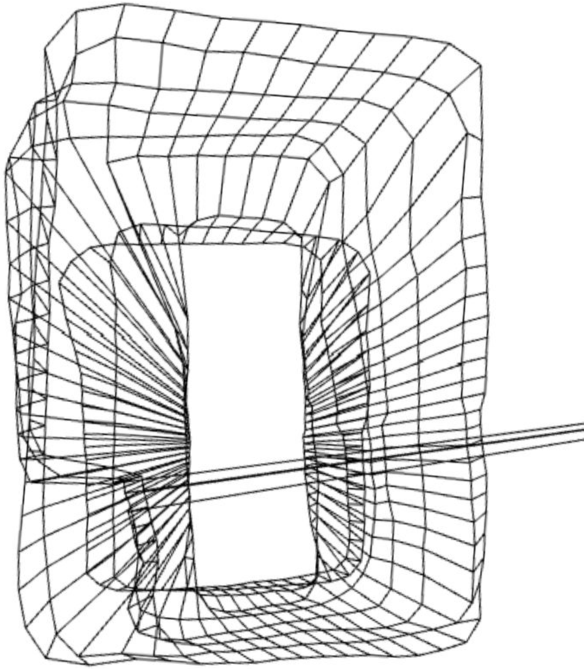
- Built on top of provided base files (e.g., I2C, UART, PLL, SysTick timer setup)
- Configured GPIO, UART (PC comm), I2C (sensor comm) and control LEDs
- Polling-based state machine with ON/OFF control via push button with active low logic
- Interrupt on PJ1 (onboard push button) allows pause/resume at any time
- 32-point scan loop:



- After full 360 degree scan:
  - Return motor CCW to home position
  - Increment x-position by default 800mm, or other user-specified amount

04

Data Visualization

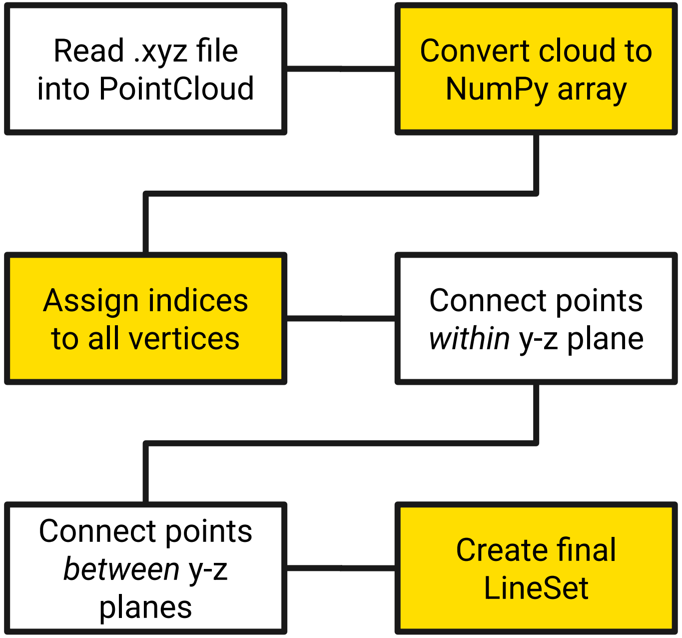


Data Visualization

Data Capture & File Creation

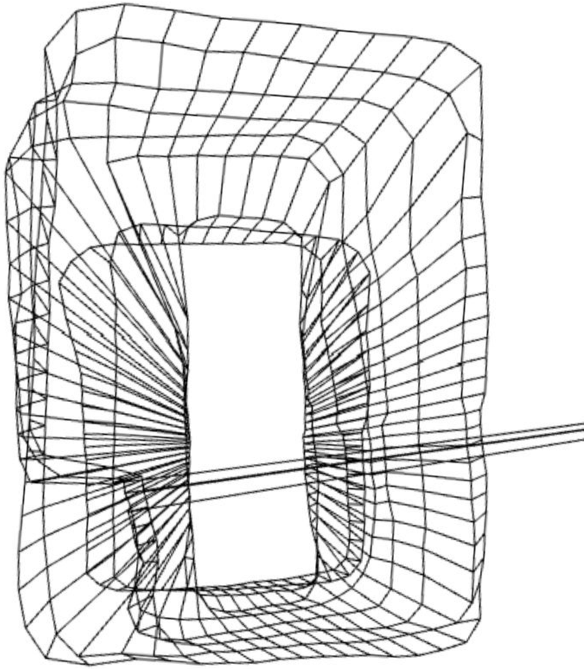
- Open **serial connection** to micro (e.g., COM4 at 115200 baud)
- Read (x, y, z) strings line-by-line over **UART**
- **Parse** coordinates
  - Split whole line string "0, 456, 29560, 45, 64" to array of individual strings ["0", "456", "29560", "45", "64"]
  - Convert x, y, z strings to ints
- Append coordinates to **.xyz file**
- Close serial port

3D Rendering with Open3D



05

Challenges



Problems Encountered & Fixed

1 Stepper Motor Cables Tangling

Only moving in CW direction caused cable entanglement → Return-to-home CCW spin after each full revolution to untwist cables

2 Sensor Boot Failing on Power-On

Rare failure in VL53L1X BootState polling loop caused infinite wait → Timeout counter & LED feedback for sensor ack; system now forces restart if sensor unresponsive

3 Missed Button Presses Due to Bounce

Sometimes input wouldn't register, requiring multiple presses → Debounce logic to confirm sustained low level before changing state

Debounce Logic Pseudocode

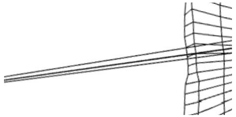
```
While state is OFF
  get PM0 input reading
  if reading is not last reading
    reset debounce timer

  if it's been longer than debounce time:
   set state to ON (PM0 button has been pressed)

  set last reading to current reading
```

Speed - Bottlenecks & Solutions

Blockers	Current Bottleneck(s)	Improvement(s)
ToF Sensor Timing Budget	100 ms ranging + 200 ms delay for ~300 ms/measurement	Switch to a faster ToF sensor (e.g., custom photodiode + ADC), reduce ranging mode distance to sub-4m
Stepper motor rotation delay	64 motor steps per 11.25deg (2048 steps/rev), with 2 ms/step (to avoid skipping)	Use higher-torque, lower-latency stepper, overlap motor step delay with sensor wait time
UART throughput	115200 bps = ~10 ms per (x, y, z) data point (~30 characters per line) → *115.2 kbps max for stable PC comms*	Transmit raw distance measures from micro to PC and perform trig calculations in Python code; move from blocking to non-blocking UART
Orthogonal displacement	Human delay to reposition device, visually align & press button ~5-10 sec	Automate orthogonal displacement with stepper-driven linear rail or actuator arm
Return-to-home CCW rotation	Full reverse rotation to untangle wires ~2 sec per rev	Remove CCW return via slip-ring or swivel mount

Resolution - Bottlenecks & Solutions

Blockers	Current Bottleneck(s)	Improvement(s)
ToF Sensor Resolution	16-bit ADC, with max quantization error ~0.061mm ($4/2^{16}$)	Use a ToF sensor with a narrower FOV or higher precision; combine multiple sensors at different angles
Angular Resolution (**y-z**)	Limited to 11.25deg (32 scans per revolution)	Increase angular sampling with smaller step size (e.g., 5.625deg for 64 points/rev) → tradeoff with speed
Linear Resolution (**x**)	Limited by manual human movement (configurable in FW, but imprecise in actuality)	Reduce x-step size; automate x-axis movement with stepper-driven linear rail for consistent fine scanning (e.g., 50mm spacing)

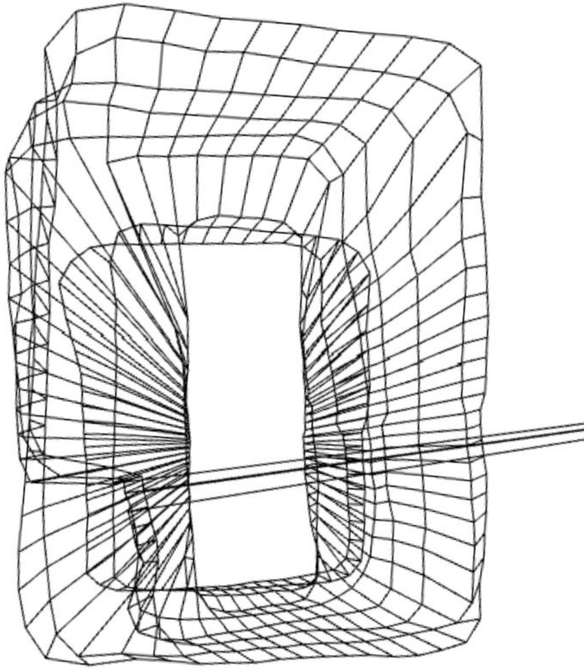


- Sharp line extending far beyond expected depth.
- An open garbage bin blocked the emitted IR pulse from reflecting back (trapping light).
- The VL53L1X sensor returned a default “max range” value of 8190mm (16-bit max).
- Easily fix in Python by discarding obvious outliers (e.g. > 4m).

Out-of-Range Artifact from Signal Loss`

06

Reflection



Lessons Learned - Then vs. Now

Then	Now
Focused on getting things to work - made design choices arbitrarily	Start by defining specs, constraints & performance targets. Ground design decisions in spec impact.
Failed to optimize system architecture (e.g., trig on MCU, tedious polling)	Constantly ask questions. Is this working in the **most efficient** way? Are failure modes accounted for? Can this scale or be reused (modular)?
Didn’t justify part selection beyond baseline requirements	Choose components carefully, considering cost, size, power, speed, resolution & system-level impact.