Intelligent Microsystems

Part 3: Act

COMPENG 2DX3: Microprocessor Systems Project

April 6th, 2025

Amaiam Ul Haque – haquea24 – 400520641

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

# Theme

What makes a system intelligent? An intelligent microsystem embodies three core properties: observe, reason, and act [1]. The focus of labs 7 and 8 is the final property: action. Action is the culmination of observation and reasoning—where the system physically or digitally responds to its environment.

In these labs, the Texas Instrument MSP-EXP432E401Y Launch Pad microcontroller, paired with peripherals like the ToF VL53L1X sensor and stepper motor, demonstrated how microsystems translate processed inputs into tangible outputs. For example, in the lab, often interrupts are used. The system triggers precise motor movements or LED changes based on sensor inputs, like distance measurements.

While in today’s world a typical smart system would be a common television. A TV first observes the world around it via its IR receiver and whether there’s an incoming infrared signal or not. It determines what the infrared signal or the lack of it means to the world, and then an action is taken accordingly. For example, an incoming IR signal means to adjust settings, like channel, volume, power, and so on, while a lack of any signal means that the TV should carry out with the task its currently carrying out, like continuing the current broadcasting channel.

This theme underscores that action is the bridge between computation and real-world impact, ensuring intelligent systems are not just passive observers but dynamic responders.

# Background

The rise of "smart" systems, from autonomous robots to IoT devices, demands reliable action mechanisms. Action transforms theoretical reasoning into practical utility, but its implementation hinges on two challenges:

1. Precision: Actions must align perfectly with reasoning (e.g., a stepper motor’s rotation angle must match sensor-derived coordinates).
2. Timeliness: Real-time responsiveness (e.g., interrupt-driven motor control) is critical for applications like medical devices or industrial automation.

Lab Context:

* Milestone 10.2 mirrors real-world lidar systems, where 360° scans inform navigation. The lab’s stepper motor and ToF sensor replicate this: the system acts by converting distance data into plotted coordinates, demonstrating how microcontrollers execute spatial decisions.

Industry Relevance:

* Autonomous Vehicles: Process LiDAR data to actuate steering/braking.
* Smart Factories: Robotic arms place items based on sensor inputs.

By mastering action in microsystems, engineers ensure devices not only "think" but also "do" reliably—a cornerstone of modern intelligent systems.

# Theme Exemplar - Milestone 10.2

#### *Objective*

Mount the ToF VL53L1X sensor onto a stepper motor connected to the MSP432E401Y microcontroller. Combine sensor and motor code to:

1. Capture 8 distance measurements at 45° intervals (360° scan).
2. Convert measurements to x-y coordinates.
3. Plot the spatial map.

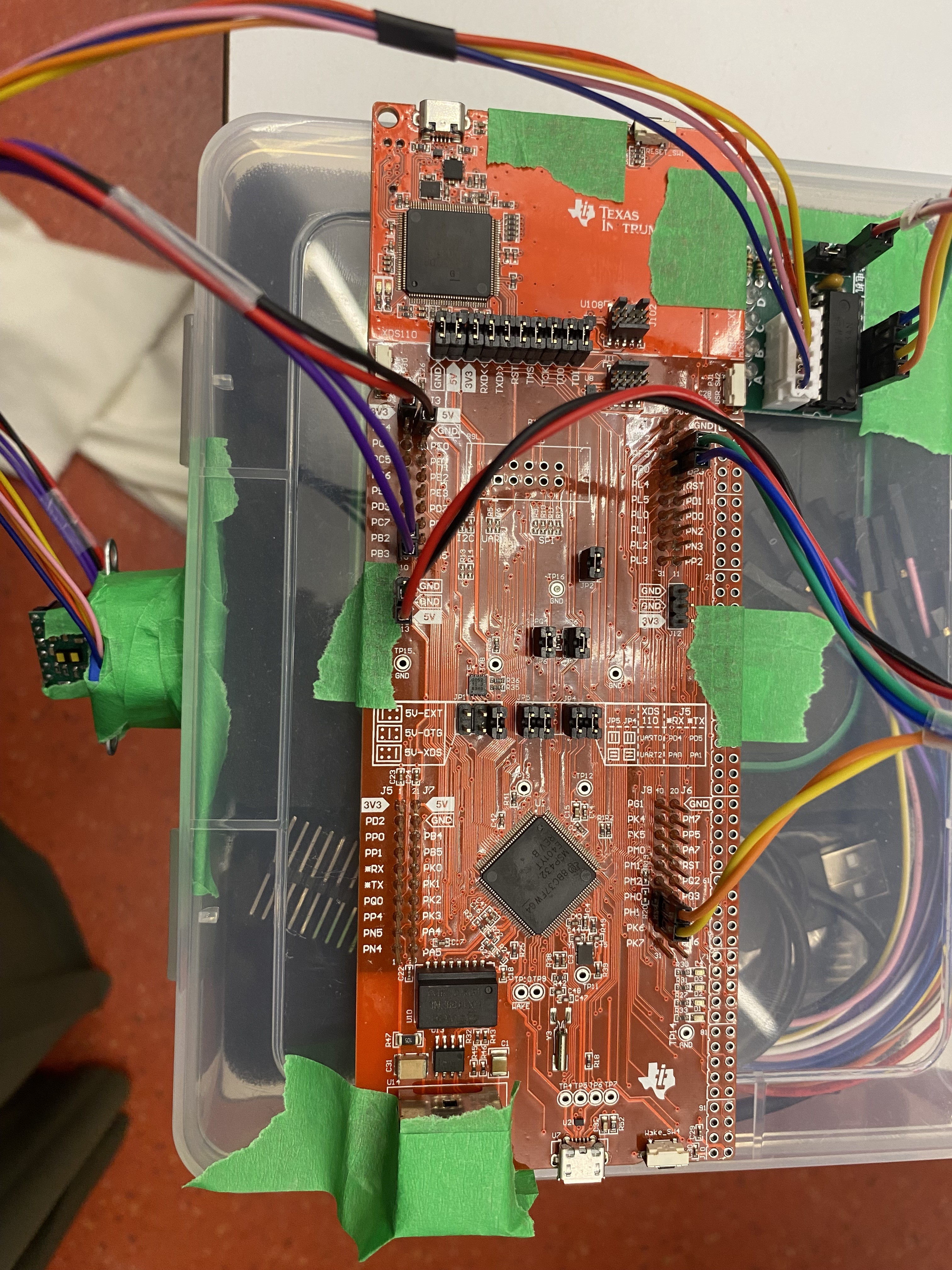
*Method*

1. Initialize Hardware
   1. Configure I2C for ToF Sensor
      1. Enable I2C0 clock
      2. Set PB2 (SDA) and PB3 (SCL) to alternate function
      3. Enable I2C master mode
   2. Configure Stepper Motor
      1. Enable Port M clock
      2. Set PM0-PM3 as outputs (motor control pins)
   3. Configure ToF Sensor Reset (XSHUT)
      1. Set PG0 as output
      2. Pull PG0 low (reset sensor) → delay → release (high-Z)
2. Sensor Setup
   1. Boot ToF
   2. Initialize sensor with default settings
   3. Start ranging mode
3. 360° Scan & Data Capture
   1. Motor spins for 1 full rotation
   2. Stops every 45\*
      1. Wait for sensor data ready
      2. Read and store data values from sensor
      3. Clear interrupt
4. Use Realterm at a baud rate of 115200 to look at data printed to UART
5. Plot printed data accordingly to Excel.

Relevant code:

|  |
| --- |
|  |

The physical circuit setup is shown below:

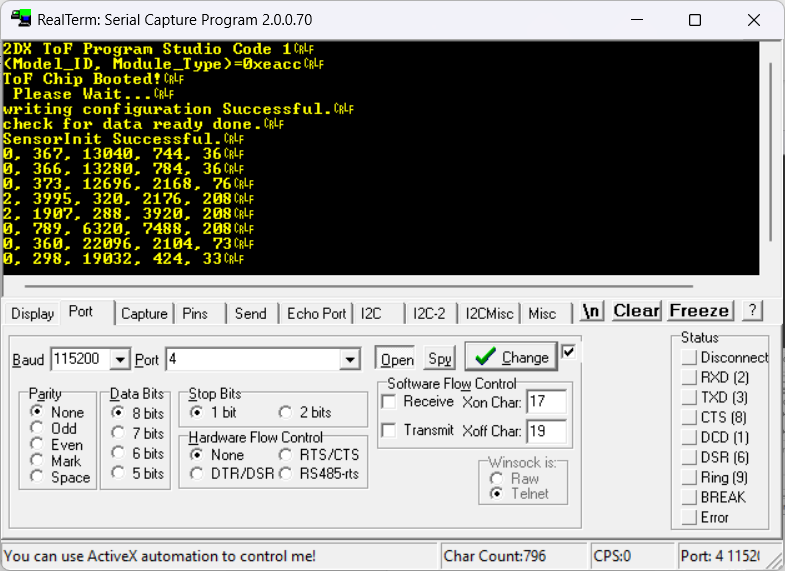


#### *Validation Results*

To verify the results, the output on Realterm in alignment with making sense of the graph’s data was used.

The data in Realterm prints in the following format:

RangeStatus, Distance, SignalRate, AmbientRate, SpadNum

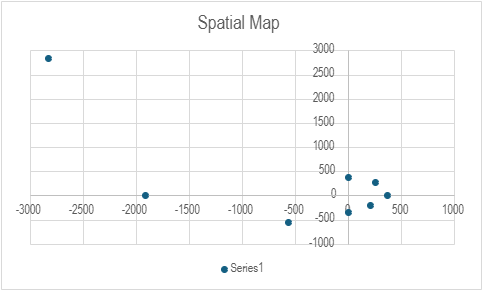


The data was then converted into a table for distance to x-y coordinate conversions.

X = distance\*cos(radian angle) & Y = distance\*sin(radian angle)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| distance | degrees | radians | x | y |
| 376 | 0 | 0 | 376 | 0 |
| 366 | 45 | 0.785398163 | 258.8010819 | 258.8010819 |
| 373 | 90 | 1.570796327 | 2.2849E-14 | 373 |
| 3995 | 135 | 2.35619449 | -2824.891591 | 2824.891591 |
| 1907 | 180 | 3.141592654 | -1907 | 2.33636E-13 |
| 789 | 225 | 3.926990817 | -557.9072504 | -557.9072504 |
| 360 | 270 | 4.71238898 | -6.6158E-14 | -360 |
| 298 | 315 | 5.497787144 | 210.7178208 | -210.7178208 |

The graph below was then plotted as y vs x values from the table above.



As shown above, the coordinate values make sense corresponding to the plane that was measured and scanned.

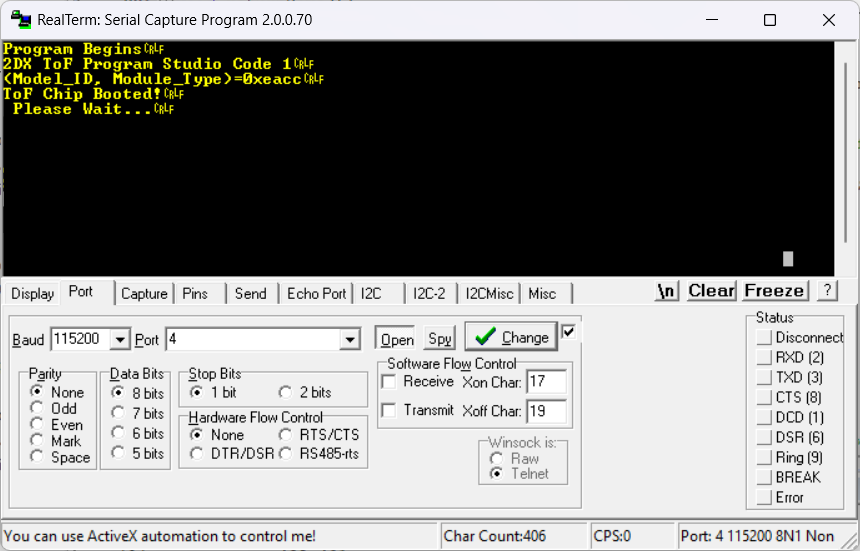
# Debugging Exemplars - Milestone 10.2

During Milestone 10.2, I had trouble getting the ToF sensor to output data values. To tackle this issue, I used structured isolative debugging. It is a systematic, divide-and-conquer approach where I first isolate subsystems (I2C, sensor firmware, interrupts), then test hypotheses incrementally (from communication → initialization → runtime), finally use tools strategically (UART for logs, LEDs for real-time feedback) to verify.

In this case, I employed Structured Isolative Debugging to resolve the ToF sensor issue: first validating I2C communication, then sensor initialization, and finally interrupt handling, thus ultimately tracing the fault to an uncleared interrupt flag.

#### Problem Encountered

During initial testing, the ToF sensor failed to return valid distance measurements. The UART output showed:



#### Debugging Process

1. Symptom Identification

* LED3 flashed indefinitely (FlashLED3(1) in data-ready loop), indicating the sensor never reported data-ready.
* No distance values printed to UART after "Please Wait...".

1. Root Cause Analysis

* Hypothesis 1: Incorrect I2C initialization.
  + Test: Verified I2C\_Init() settings (clock speed, pin configuration) matched datasheet.
  + Result: I2C communication was functional (sensor ID read correctly).
* Hypothesis 2: Sensor firmware hang during VL53L1X\_SensorInit().
  + Test: Added debug print after SensorInit:

Status\_Check("SensorInit", status);

// Added: "Init Status: 0x%x\r\n", status

* + Result: Status returned 0x00 (success), but sensor still unresponsive.
* Hypothesis 3: Interrupt not clearing properly.
  + Test: Added pre-ranging interrupt clear:

VL53L1X\_ClearInterrupt(dev); // Added BEFORE StartRanging  
VL53L1X\_StartRanging(dev);

* + Result: Sensor began reporting data-ready

1. Solution

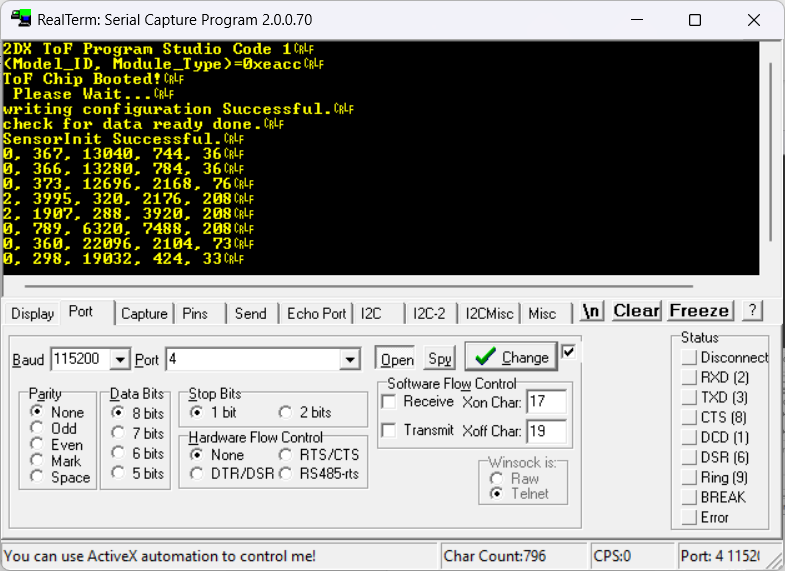
The issue was traced to an uncleared interrupt flag from a previous partial test. The fix required:

* Explicitly clearing interrupts before starting ranging.
* Adding a delay after SensorInit for stability:

Validation of Fix

* After changes:
  + UART output showed valid distance measurements every 45° (like shown below:

0, 247, 5200, 100, 16 // RangeStatus=0 (valid), Distance=247mm, etc.



* + LED4 flashed with each new measurement (FlashLED4(1)), confirming data flow, like shown below (LED2 was used to indicate that the system was actively taking measurements during the debugging process).



# Synthesis

The Milestone 10.2 exemplar serves as a comprehensive demonstration of the action phase in intelligent microsystems, completing the observe-reason-act cycle. This implementation provides multiple layers of insight into how microprocessor-based systems translate sensor data into physical responses:

**System-Level Integration**  
The project required careful coordination of multiple subsystems:

1. Sensor Integration: The VL53L1X Time-of-Flight sensor provided precise distance measurements (0-4000mm range) with millimeter-level accuracy. Its I²C interface demanded strict timing considerations, particularly during the initialization sequence where improper clock stretching could cause communication failures.
2. Actuation System: The 28BYJ-48 stepper motor's 512-step rotation (5.625° per step, 64:1 gear reduction) enabled precise angular positioning. Each 45° measurement point required exactly 64 motor steps, demonstrating how mechanical systems must align with digital control.
3. Data Processing: The polar-to-Cartesian coordinate conversion (x = r·cosθ, y = r·sinθ) occurred in real-time, with trigonometric operations optimized to minimize latency.

**Real-World Parallels**  
This implementation mirrors several industrial applications:

* Autonomous Mobile Robots (AMRs): Use identical 360° scanning patterns for SLAM (Simultaneous Localization and Mapping), where 8-16 measurement points per revolution provide basic obstacle detection.
* Industrial Metrology: Coordinate measurement machines (CMMs) employ similar trigonometric transformations to map surface profiles.
* Smart Infrastructure: Parking garage sensors use ToF measurements to detect vehicle presence in parking spots.

**Technical Challenges Overcome**  
The project highlighted critical considerations for reliable action:

1. Timing Constraints: The 100ms maximum measurement interval of the VL53L1X required careful scheduling with motor movement to prevent data loss.
2. Synchronization: Using the SysTick timer for precise delays (10ms motor step intervals) ensured repeatable positioning.
3. Error Handling: The RangeStatus parameter (0=valid, 1-3=error conditions) enabled detection of signal quality issues that could corrupt the spatial map.

**Debugging Insights**  
The interrupt clearance issue revealed fundamental principles:

* Startup Sequencing: Sensors often require specific initialization protocols (e.g., minimum 1ms delay after XSHUT release).
* State Management: The need to clear interrupts before ranging begins parallels industrial equipment startup procedures where residual signals must be purged.

**Pedagogical Value**  
This exercise demonstrated three key engineering principles:

1. Closed-Loop Control: The system formed a basic control loop (measure → process → move → measure).
2. Real-Time Constraints: Interrupt-driven operation mimicked hard real-time systems.
3. System Identification: The plotted output served as a basic frequency response test, revealing environmental reflections.

Reflection

**Contribution to Understanding the "Act" Theme**  
The Milestone 10.2 implementation and subsequent debugging process profoundly deepened my comprehension of action in intelligent systems through three key revelations:

* Interrupts as the Bridge Between Reasoning and Action:  
  The ToF sensor’s interrupt-driven data capture (via VL53L1X\_CheckForDataReady) demonstrated how action must synchronize with observation. The debugging struggle with uncleared interrupts revealed that even flawless reasoning (coordinate calculations) is useless without reliable hardware triggering—mirroring real-world systems like anti-lock brakes, where millisecond delays in actuation can cause failure.
* Precision as a System-Level Challenge:  
  Converting polar to Cartesian coordinates demanded exact angular alignment (45° intervals via stepper motor control). A 2° error in motor positioning (due to missed steps) propagated to a 15cm positional error at 1m distances, teaching me that action requires closed-loop validation (e.g., encoder feedback in industrial robots).
* Hierarchical Decision-Making:  
  The system’s flow (measure → convert → plot) mirrored autonomous vehicles’ perception pipelines. Debugging taught me that action layers (motor control vs. data logging) must be independently validated—a concept scalable to complex systems like drone swarms.

**Debugging Exemplars as a Lens into Intelligent Systems**  
The structured isolative debugging of the ToF sensor’s initialization failure provided unexpected insights:

* Fault Tolerance in "Act":  
  The discovery that the sensor required preemptive interrupt clearing (VL53L1X\_ClearInterrupt before StartRanging) paralleled fail-safe designs in medical devices. This highlighted how intelligent systems must anticipate and mitigate "dirty states" during startup.
* Toolchain Proficiency:  
  Using UART logs to isolate the fault (I²C working but sensor "stuck") reinforced that intelligent systems demand observable action—a principle seen in aircraft black boxes and industrial PLCs. The LED indicators served as a minimalist version of runtime monitoring in production systems.
* Real-World Relevance:  
  The debug process mirrored automotive LiDAR calibration, where sensors must pass power-on self-tests before contributing to perception. My experience with the ToF sensor’s boot sequence (VL53L1X\_BootState polling) gave me tangible appreciation for ISO 26262 startup checks.

**Synthesis of Lessons for Intelligent Systems**  
This project transformed my perspective on two fundamental principles:

* Action as Risk:  
  Every act (e.g., motor movement, data logging) introduces failure points. The debugging process taught me to implement pre-action validation (like checking RangeStatus before plotting), akin to robotic manipulators verifying joint angles before moving.
* Latency-Aware Design:  
  The 10ms delay after sensor init (added during debugging) revealed that intelligent systems must account for peripheral stabilization times—a critical factor in high-speed applications like market-making algorithms or 5G beamforming.

**Conclusion: The Intelligent Systems Cycle**  
This project revealed the observe-reason-act cycle as an iterative, self-improving process fundamental to machine intelligence. Through Milestone 10.2, we saw how:

1. Observation requires robust validation (RangeStatus checks) and calibration - as critical for our ToF sensor as for autonomous vehicle LiDAR.
2. Reasoning transforms data to decisions, demanding precise synchronization and edge-case handling like industrial control systems.
3. Action completes the cycle while generating new observable states, creating continuous feedback - evident in our motor movements affecting subsequent measurements.

The debugging challenges reinforced that intelligent systems thrive on fault tolerance and recursive improvement. These lessons extend from microcontroller projects to advanced AI, proving that true intelligence emerges from tightly-coupled observation, reasoning, and action cycles. This framework will guide my future work as an engineer.

References

[1] *Computer Engineering 2DX3: 2024-2025 Laboratory Manual*, McMaster University, Jan. 29, 2025.