Victor Perez Contreras
Tae M. Huh
ECE 118
April 26, 2025

Part 4 – Sensor System Design

Goal: Choose two distinct sensor types (excluding serial-based prepackaged modules that provide digital counts through serial communication).

Sensor 1: Force Sensor

Model and Vendor

Model: FSR402

Vendor: Interlink Electronics

Sketch and Description

Sketch:

Description: Sensor that detects force applied to it. The sensor is added to a circuit with a voltage divide with a pull-down resistor. As the force increased in the sensor, resistance decreases. The reading can then be read by the UNO32′s analog input.

RC Low-papss filter is added between the voltage divider and the ADC input:

$R_2 = 1k\Omega$

$C_1 = 0.1\mu F$

Noise Sources

Sources:

1) Vibraition can cause flucatating reading

2) Electrical noise

Mitigation:

1) Software filtering (averages)

2) Calibration

Data Sheet Analysis

Resistance range: > 10MΩ (no force) to 1kΩ (full force)

Force sensitivity range: 0.2N to 20N

With the 10kΩ pull-down resistor in the voltage divider:

- At no force: VOUT ≈ 0V (FSR resistance very high)
- At light force (0.2N): VOUT ≈ 0.5V (FSR ≈ 50kΩ)
- At medium force (5N): VOUT ≈ 2.5V (FSR ≈ 10kΩ)
- At high force (20N): VOUT ≈ 4.5V (FSR ≈ 1kΩ)

Psuedocode

```
def monitorDrawForce():
    # Initialize variables
    int SAMPLE_COUNT = 10
    int MIN_DRAW_THRESHOLD = 100    # ADC value
    int MAX_DRAW_THRESHOLD = 900    # ADC value
    int RELEASE_THRESHOLD = 50      # ADC value

    int forceReadings[SAMPLE_COUNT]
    bool drawDetected = false
    bool releaseDetected = false

    # Calculate average force
    sum = 0
    for i = 0 to SAMPLE_COUNT-1:
        sum += forceReadings[i]
    averageForce = sum / SAMPLE_COUNT

    # Check if draw has started
    if averageForce > MIN_DRAW_THRESHOLD && !drawDetected:
        drawDetected = true
        signalDrawStarted()

    # Check if draw has reached maximum
    if averageForce > MAX_DRAW_THRESHOLD:
        signalMaximumDraw()

    # Check if release has occurred
    if drawDetected && averageForce < RELEASE_THRESHOLD:
        releaseDetected = true
        drawDetected = false
        maxForce = 0
        signalReleaseDetected()

    # Update the current state
    updateState {
        isDrawing: drawDetected,
        currentForce: averageForce,
        maximumForce: maxForce,
        released: releaseDetected
    }
```

Sensor 2: Distance Sensor

Model and Vendor

Model: VL53L1X

Vendor: STMicroelectronics

Sketch and Description

Description: operates at a 3.3V level, similar to the UNO32 and PIC32. The sensor communicates using I2C protocol, requiring SCL and SDA connectors.

Pull-up resistors (4.7kΩ) are required for the I2C lines to ensure proper communication. A 0.1 µF decoupling capacitor should be placed close to the VDD pin of the sensor to filter power supply noise.

Sketch:

Noise Sources

Potential Noise Sources:

1) Power supply fluctuations

2) EMI from motors and actuators

3) Ambient light interference (IR)

Mitigation Methods:

1) Power Supply Filtering:

Add a 10µF capacitor in parallel with a 0.1µF capacitor close to the sensor power pin

2) EMI Shielding:

- Keep I2C lines short and away from high-current paths
- Add ferrite beads on power and signal lines
- Use shielded cables for connections if possible

3) Optical Isolation:

Mount the sensor in a recessed housing to reduce ambient light interference

Data Sheet Analysis

According to the VL53L1X datasheet:

Measurement range: Up to 4 meters

Accuracy: ± 2% typical

Resolution: 1mm

Output: Digital via I2C interface (16-bit distance value in mm)

The UNO32 reads digital values directly through I2C, so ADC resolution is not applicable. The 16-bit distance values provide millimeter-level precision, which is more than sufficient for target distance determination in an indoor archery setting.

The sensor can operate in different modes:

- Short range: Up to 1.3m (higher accuracy)
- Medium range: Up to 3m
- Long range: Up to 4m (lower accuracy)

For an indoor archery robot, the medium range mode provides a good balance of accuracy and range. With 1mm accuracy at target distances of 1-3m, this provides sufficient precision for calculating firing angles to relatively far objects.

Psuedocode

```
int detectTarget(){
  const int SAMPLE_COUNT = 5;
  const int TARGET_PRESENT_THRESHOLD = 2000  // mm (2 meters)
  const int DISTANCE_CHANGE_THRESHOLD = 50   // mm

  int distanceReadings[SAMPLE_COUNT]
  int averageDistance = 0
  int previousDistance = 0
  bool targetDetected = false

  // Take an average of the distance
  for (int i = 0; i < SAMPLE_COUNT-1; i++) {
    startMeasurement()
    waitForDataReady()
    distanceReadings[i] = readDistance()
  }

  // Calculate average distance
  int sum = 0
  for (int i = 0; SAMPLE_COUNT-1; i++) {
    sum += distanceReadings[i]
  }
  averageDistance = sum / SAMPLE_COUNT


  // Check if object is within expected target range
  if (averageDistance < TARGET_PRESENT_THRESHOLD) {
    targetDetected = true;
  }

  // Check if target has moved
  if abs(averageDistance - previousDistance) > DISTANCE_CHANGE_THRESHOLD {
    signalTargetMovement(averageDistance)
  }

  previousDistance = averageDistance

  // Return detection result
  return {
    detected: targetDetected,
    distance: averageDistance
  }
  }
```

Sources

Sensor 1: https://cdn.sparkfun.com/assets/8/a/1/2/0/2010-10-26-DataSheet-FSR402-Layout2.pdf

Sensor 2: https://www.st.com/resource/en/datasheet/vl53l1x.pdf