FPGA RADAR PROJECT

JOONHO JANG, CAMERON COWAN

TABLE OF CONTENTS

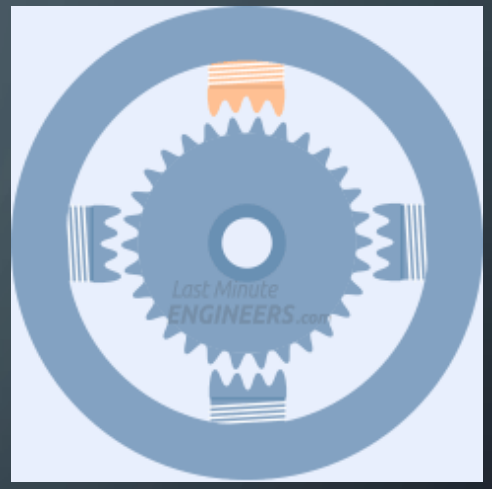
HIGH – LEVEL PROJECT OVERVIEW

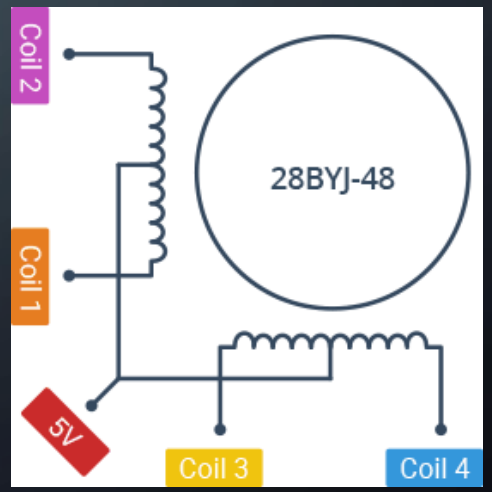
- Project: Radar Sensor
  - Extensively uses external hardware
  - Focuses on signal analysis, control signal generation, and component synchronization
- Goal:
  - 360° distance sensor (“radar”), XY position displayed through VGA
- 3 Major Submodules
  1. Ultrasonic Sensor Control (Milestone 1)
  2. Stepper Motor Control (Milestone 1 & 2)
  3. VGA Control (Milestone 2)

28BYJ-48 – 5V STEPPER MOTOR

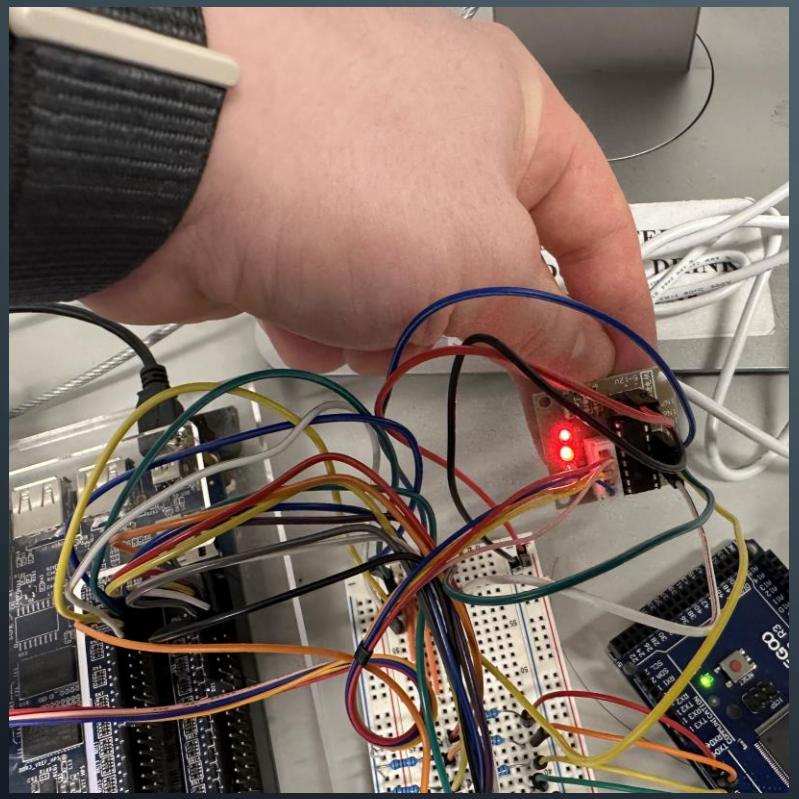
- Each coil corresponds to different wires
- 3 wires have logic LOW & 1 wire has logic HIGH
- Motor rotates by alternating each wire to have logic HIGH
- Full rotation requires 64 steps

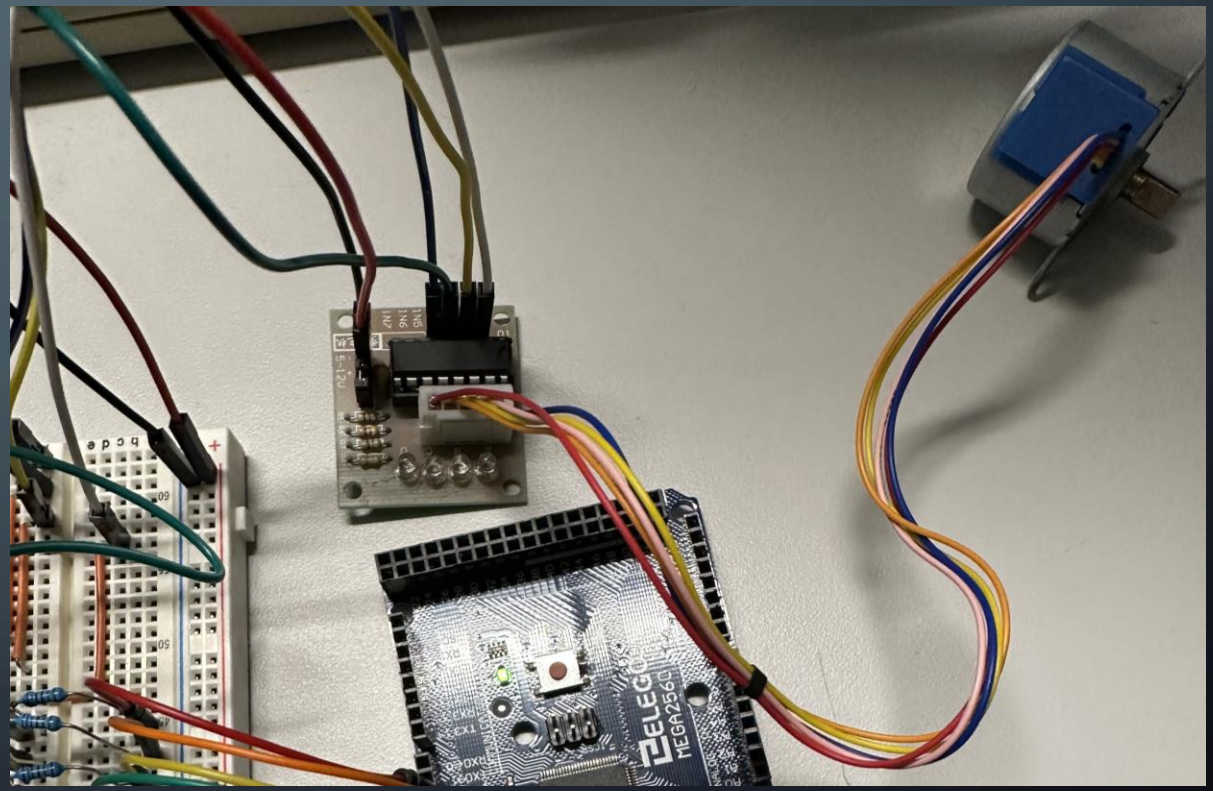
- DE1-SoC Board signal → Motor driver circuit → Motor
  - Logic High (1) → 3.3V → 5V
  - Logic Low (0) → 0V → 0V



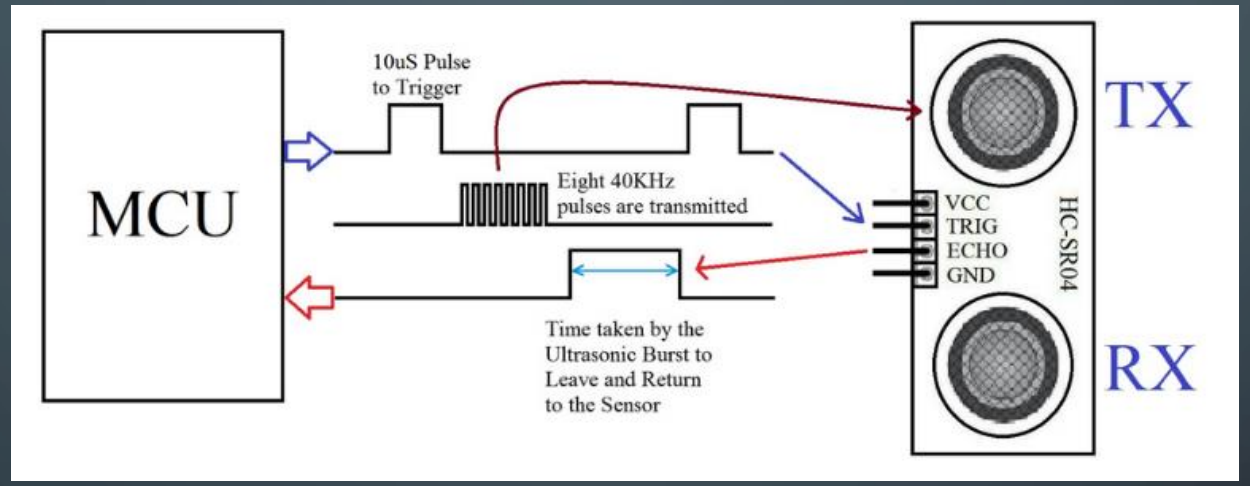


28BYJ-48 – 5V STEPPER MOTOR



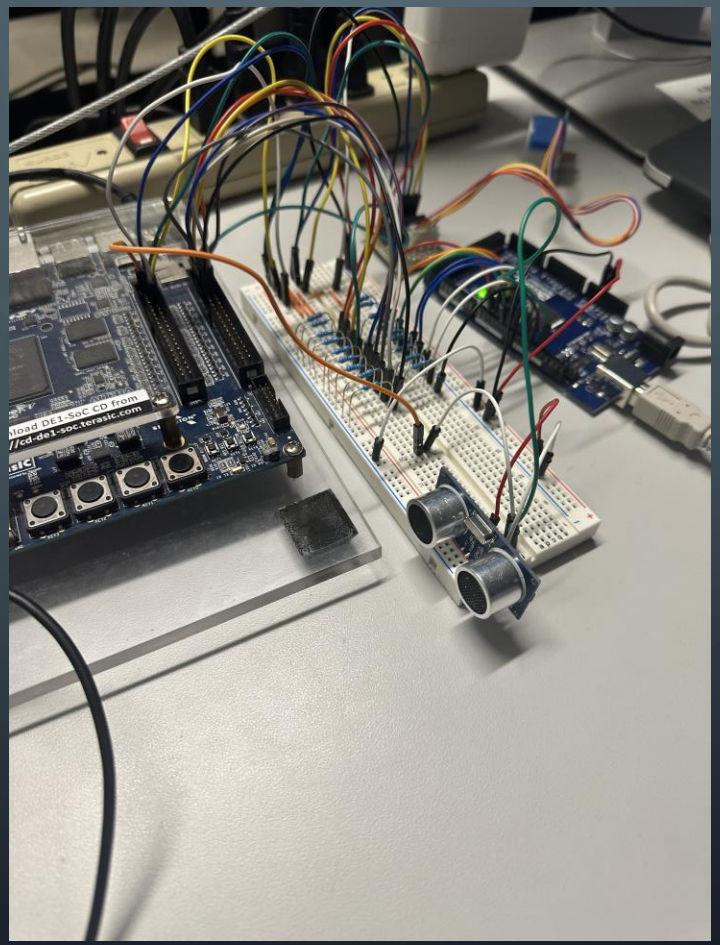


HR-SRO4 ULTRASONIC SENSOR



1. Trigger: 5V pulse powers the ultrasonic sensor to output eight 40kHz pulses
2. Echo: Maintains high voltage until RX receives the eight 40kHz pulses (PWM)

HR-SR04 ULTRASONIC SENSOR



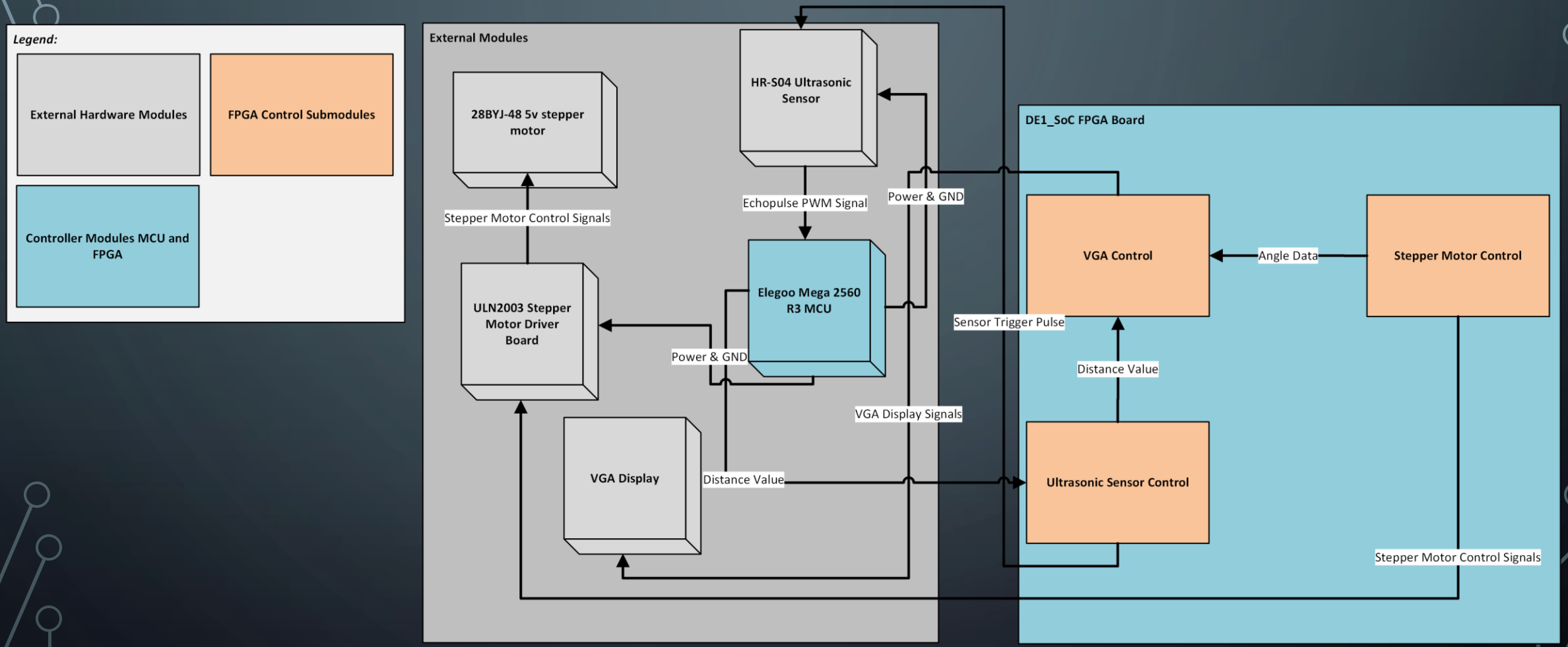
EXTERNAL HARDWARE OVERVIEW

Hardware	Input	Output	Function
Ultrasonic Sensor	• Trigger control pulse	• Echolocation PWM Signal	• Send out sonar pulses based on trigger signal • Receive sonar pulses, output pulse width modulated (PWM) signal
Stepper Motor	• Motor control signals (4 signals)	• N/A	• Transition between steps based on control signals
MCU	• DC power input • Echo pulse signal	• Distance value (9 bit wide signal) • 5V power • GND	• Provide power to motor and sensor • Receive the echo pulse signal • Convert echo signal into distance • Transmit distance to FPGA

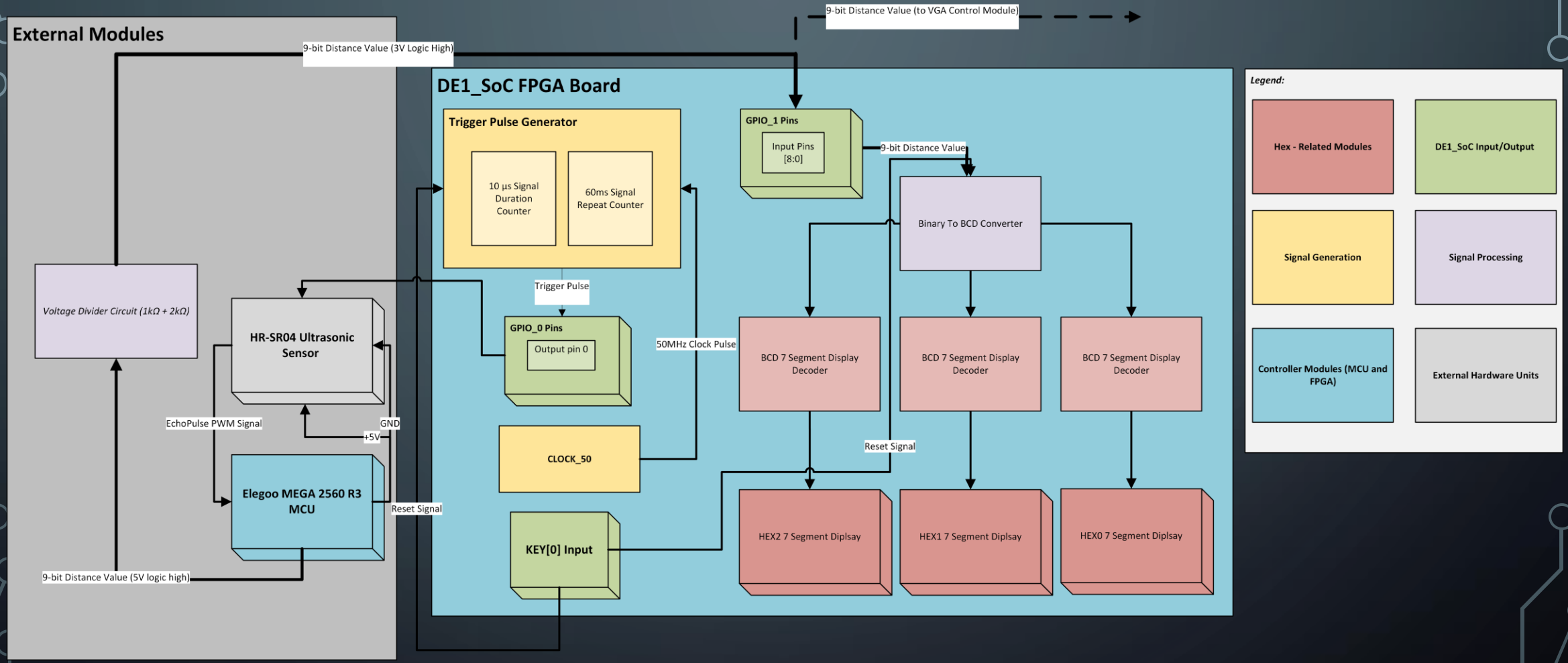
SUBMODULE OVERVIEW

Submodule	Input (FPGA)	Output (FPGA)	Function
Ultrasonic Sensor Control	• Distance value (9 bits wide – max 400, encoded in cm)	• Trigger pulse • HEX displays	• Extract distance value from ultrasonic sensor hardware • Display distance value on HEX0 to HEX1
Stepper Motor Control	• N/A	• Stepper motor control signals (4 bits wide, sent to motor driver)	• Control CCW movement of stepper motor
VGA Control	• N/A	• VGA output signal (sent to VGA adapter)	• Display cartesian position on external monitor

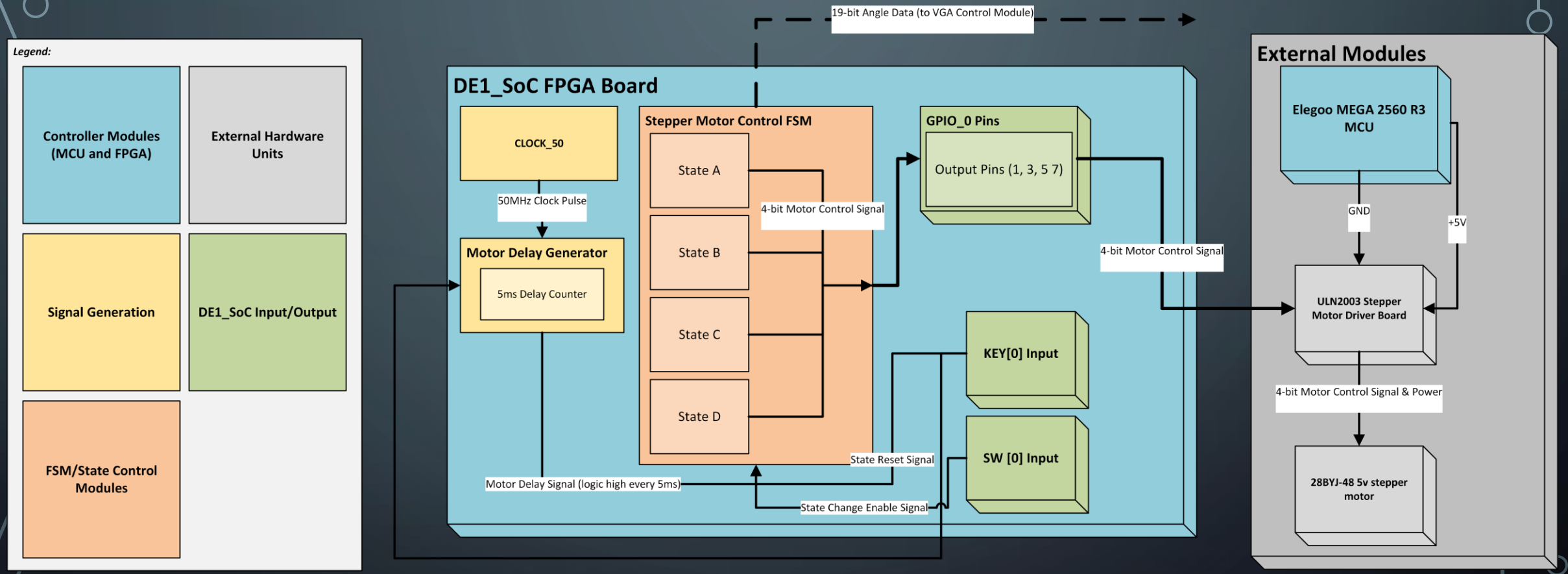
BLOCK DIAGRAM: HIGH-LEVEL OVERVIEW



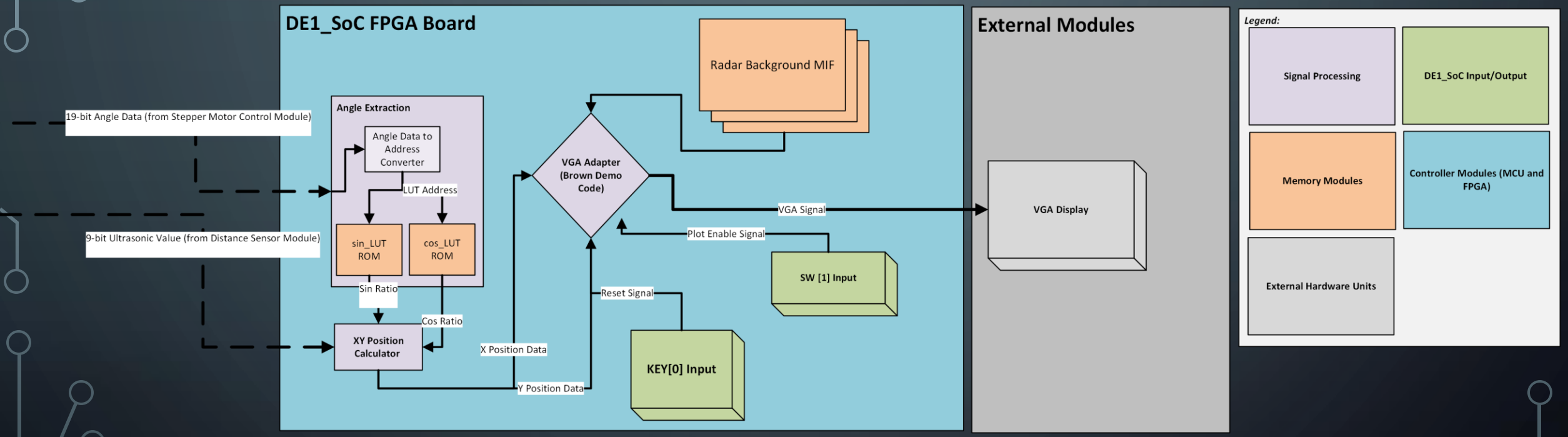
BLOCK DIAGRAM: ULTRASONIC SENSOR



BLOCK DIAGRAM: STEPPER MOTOR



BLOCK DIAGRAM: VGA CONTROLLER



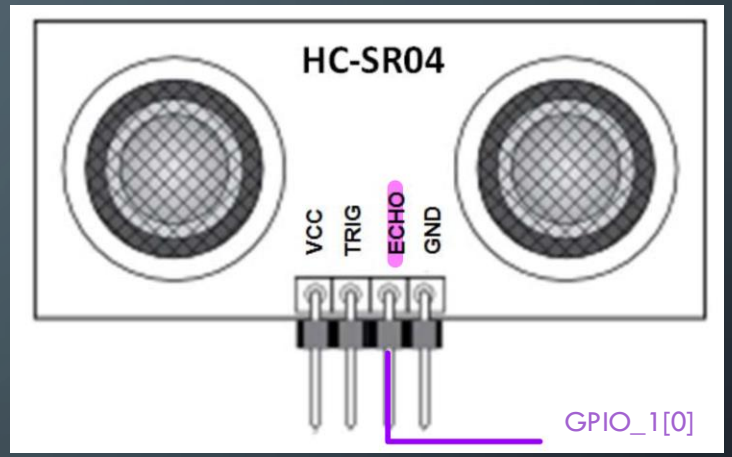
PROBLEM 1: DISTANCE CALCULATION

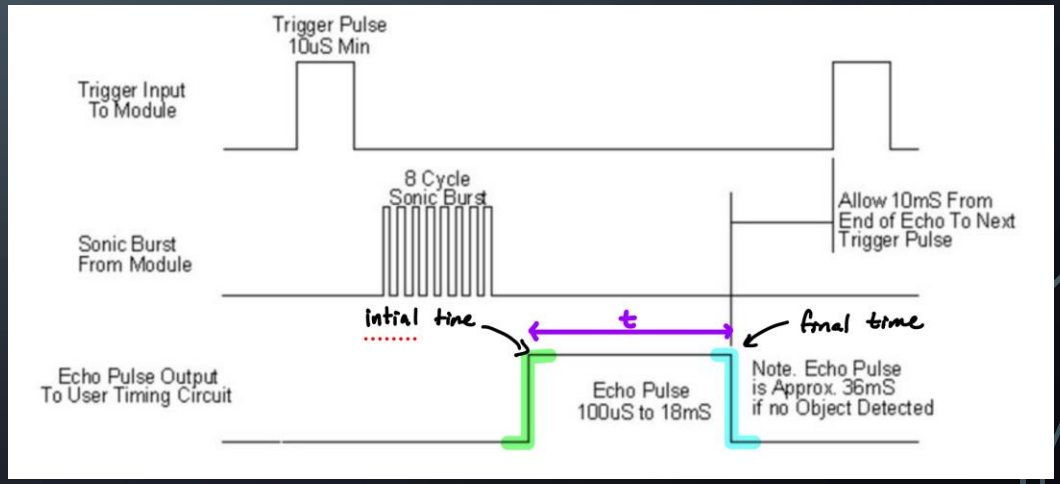
Initial Setup: Echo pin connected to GPIO_1[0]

Problem: $t = 0$ for all time

1. On posedge echoPulse → log initialTime
2. On negedge echoPulse → log finalTime
3. $t = finalTime - initialTime$

$$d(cm) = \frac{t\ (\mu s)}{58}$$





PROBLEM 1: DISTANCE CALCULATION

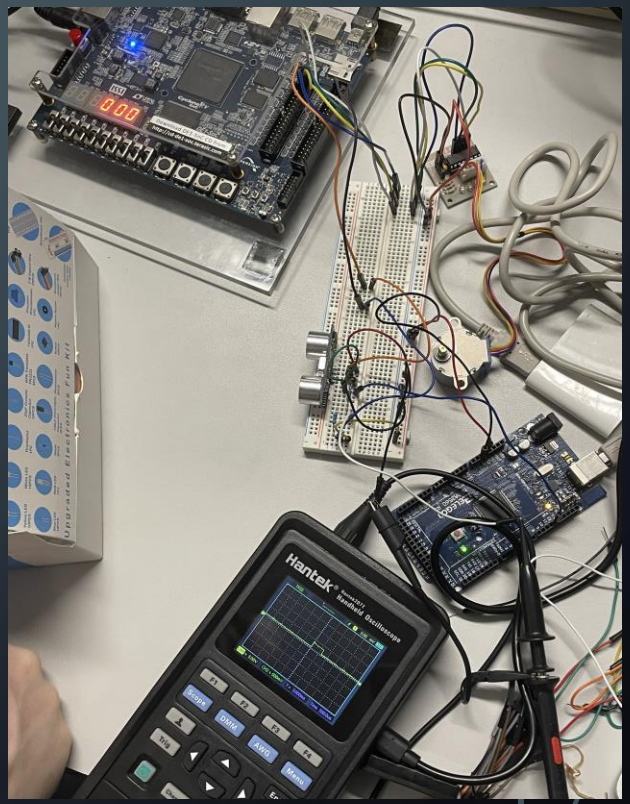
Debugging Method:

1. Setup HEX[2:0] displaying the distance (cm)
2. Probe signal sent to GPIO_1[0] using oscilloscope
3. Program MCU (Mega2560) to print distance (cm) in terminal

Conclusion:

1. GPIO Pins will not interpret echo pulse signals correctly

```
#include <Arduino.h>

int echoPin = 12;
long duration, cm;

void setup() {
  Serial.begin(9600);
  pinMode(echoPin, INPUT);

}

void loop() {
  // Read echoPin pulse duration
  duration = pulseIn(echoPin, HIGH);
  cm = (duration / 2) / 29.1;
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();

  // Delayed 250ms for visual readability
  delay(250);
}
```



PROBLEM 1: DISTANCE CALCULATION

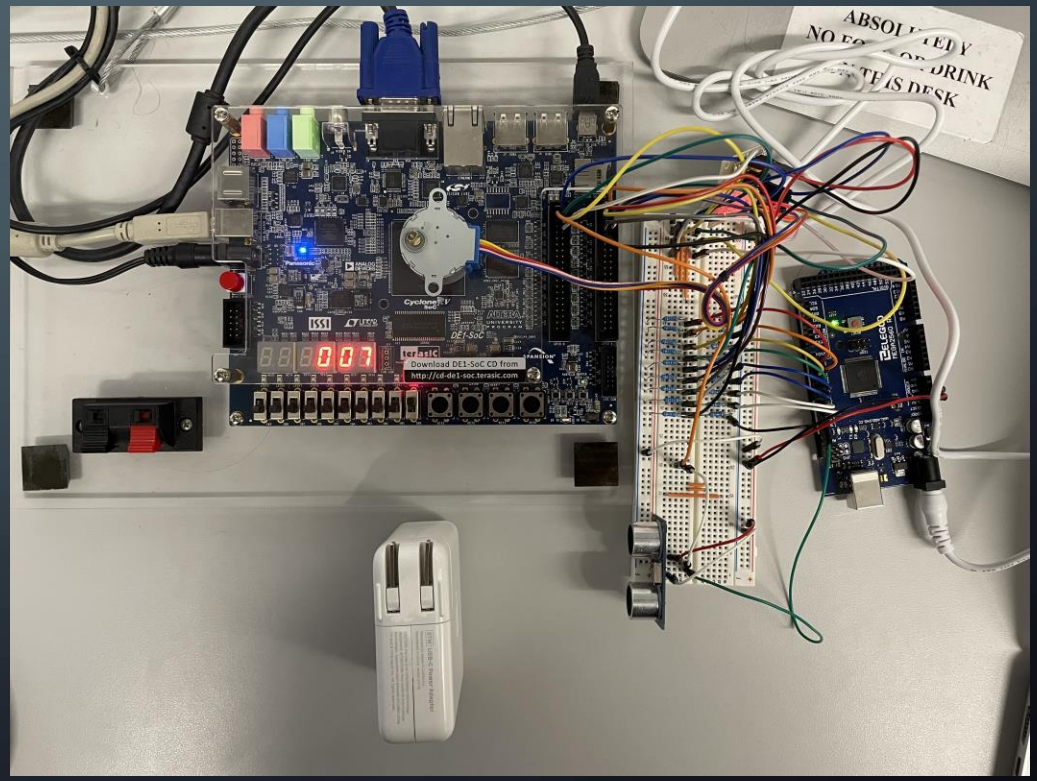
Solution: Program MCU (Mega2560) to convert distance (cm) into 9bit binary represented through pin 2~10

main.cpp | platformio.ini | PIO Home

src > main.cpp > ...

```cpp
#include <Arduino.h>

int echoPin = 12;
long duration, cm;
int numBits = 9;
int de1Soc_distance[9] = {2, 3, 4, 5, 6, 7, 8, 9, 10};

void setup() {
  Serial.begin(9600);
  pinMode(echoPin, INPUT);

  for (int i = 0; i < numBits; i++) {
    pinMode(de1Soc_distance[i], OUTPUT);
  }
}

void loop() {
  // Read echoPin pulse duration
  duration = pulseIn(echoPin, HIGH);
  cm = (duration / 2) / 29.1;
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();

  // Send bit signals to each wire
  for (int i = 0; i < numBits; i++) {
    digitalWrite(de1Soc_distance[i], (cm >> i) & 0x01);
  }

  // Delayed 250ms for visual readability
  delay(250);
}
```

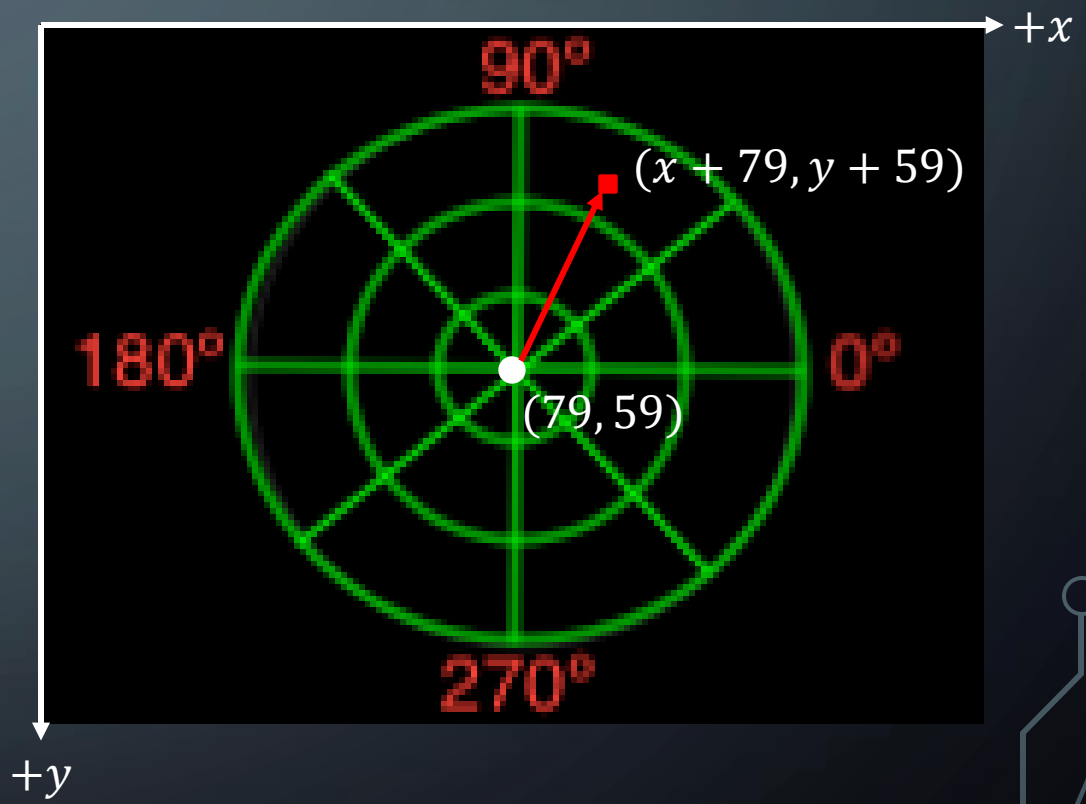
```
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
10cm
7cm
7cm
10cm
8cm
10cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
7cm
10cm
```



PROBLEM 2: TRIGONOMETRY WITH VERILOG

Initial Setup: Use trigonometry using distance (cm) & angle (degrees) to find (x, y) values

Problem: Verilog doesn't have trigonometry operations in their math library



PROBLEM 2: TRIGONOMETRY WITH VERILOG

Concept:

Input Address ← angle θ

Output Value → $\sin(\theta)$ or $\cos(\theta)$

Method:

- Input Address ← Motor's step count
  - $360°/64\text{Steps} = 5.265°/1\text{Step}$
  - Address = $(\theta * 1000)/5265$
- Output Value → $\sin(\theta) * 100$ or $\cos(\theta) * 100$
  - Accuracy to two decimal place
  - Signed value (eg. 412 ⇔ -1)

```
DEPTH = 64;
WIDTH = 9;
ADDRESS_RADIX = DEC;
DATA_RADIX = DEC;
CONTENT BEGIN
    0 : 100;
    1 : 100;
    2 : 98;
    3 : 96;
    4 : 92;
    5 : 88;
    6 : 83;
    7 : 77;
    8 : 71;
    9 : 63;
    10 : 56;
    11 : 47;
    12 : 38;
    13 : 29;
    14 : 20;
    15 : 10;
    16 : 0;
    17 : 502;
    18 : 492;
    ...
    59 : 88;
    60 : 92;
    61 : 96;
    62 : 98;
    63 : 100;
END;
```

Cos MIF Files

```
DEPTH = 64;
WIDTH = 9;
ADDRESS_RADIX = DEC;
DATA_RADIX = DEC;
CONTENT BEGIN
    0 : 0;
    1 : 10;
    2 : 20;
    3 : 29;
    4 : 38;
    5 : 47;
    6 : 56;
    7 : 63;
    8 : 71;
    9 : 77;
    10 : 83;
    11 : 88;
    12 : 92;
    13 : 96;
    14 : 98;
    15 : 100;
    16 : 100;
    17 : 100;
    18 : 98;
    ...
    59 : 465;
    60 : 474;
    61 : 483;
    62 : 492;
    63 : 502;
END;
```

Sin MIF Files

PROBLEM 2: TRIGONOMETRY WITH VERILOG

Solution: Create memory blocks that contains values for sin and cos for specific angles

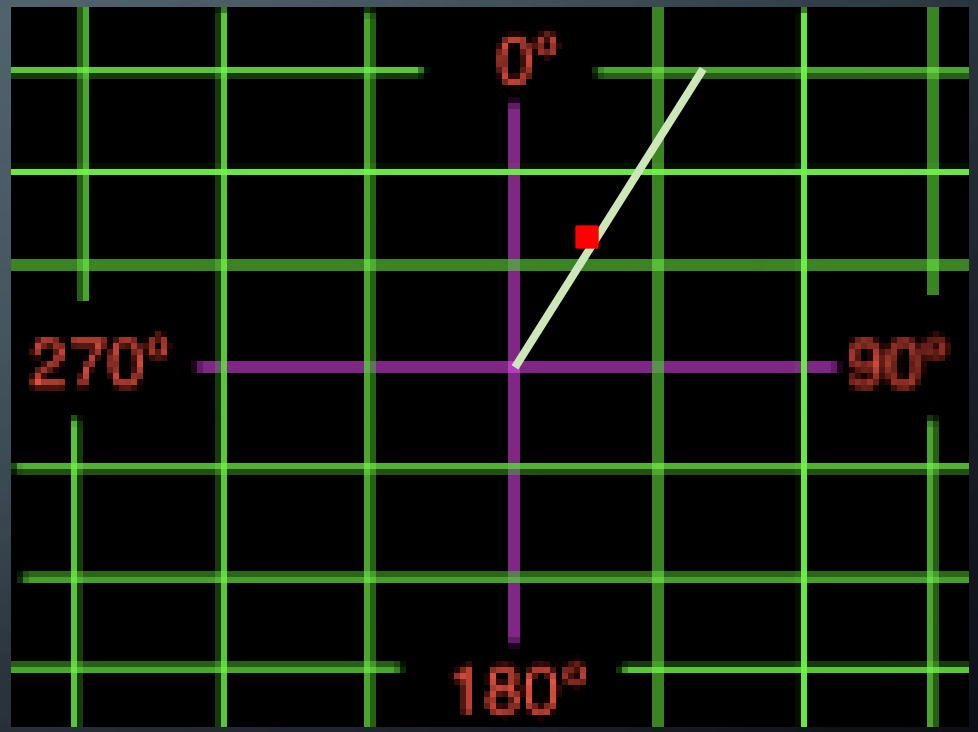
```
// Module that extracts precomputed angle fractions from lookup table
module angleExtraction (clock, sinOut, cosOut, angleCounter);
    input [18:0] angleCounter;
    input clock;
    output signed [8:0] sinOut;
    output signed [8:0] cosOut;

    // Extracting address value
    wire [5:0]address;
   assign address = angleCounter / 5625;

    // Just 0 for data in and write enable as only need to extract data
    wire [8:0] garbageDataIn = 8'b0;
    wire writeEnable = 1'b0;

    // Instantiating memory modules
    sin_LUT sin_mem (address, clock, garbageDataIn, writeEnable, sinOut);
    cos_LUT cos_mem (address, clock, garbageDataIn, writeEnable, cosOut);

    // (address, clock, dataIn, wren, output) - form for instantiating sin and cos LUT

endmodule
```

FUTURE IMPROVEMENTS

1. Mount the ultrasonic sensor to the motor
2. Improve reset functionality
   - **Removing all printed pixels**
   - **Readjust motor to an initial position (ie 0°)**

3. Scale the grid lines to ultrasonic sensor's max distance

4. Visual indicator of radar's current angle



GROUP DISTRIBUTION

*All Verilog code written synchronously. All project work done concurrently in lab conditions.

Cameron Cowan	Joonho Jang
1. Ultrasonic sensor signal conversion logic 2. Ultrasonic sensor distance calculation logic 3. FPGA, MCU, sensor interfacing 4. Ultrasonic sensor input logic 5. Stepper motor control refinement 6. VGA output .mif file generation 7. VGA demo code interfacing 8. VGA pixel printing logic 9. Angle extraction and trigonometric calculation logic	1. Ultrasonic sensor block diagram 2. Stepper motor control block diagram 3. Microcontroller debug code 4. Stepper motor control wiring (voltage step down, driver connections) 5. Breadboard configuration 6. VGA adapter demo code usage 7. XY position calculation logic 8. ModelSim configuration 9. Look up table logic