Name: Duy Khang Pham

ASU ID: 1224814539

Project Obstacle Avoidance - Distance Sensor

1. Description:

This project aims to develop a robot capable of accurately detecting objects and avoiding obstacles using an ultrasonic distance sensor. The robot intelligently navigates by continuously scanning its surroundings, making turning decisions when an obstacle is detected, and following a pre-defined path or environment layout. The implementation combines sensor feedback with servo rotation and motor control to execute real-time adjustments, ensuring reliable obstacle avoidance and autonomous movement.

2. Implementation:

a. Hardware Components:

The hardware for this project includes:

- FRDM KL46Z Board
- Breadboard
- Battery holder with rechargeable batteries
- Motors with Encoders and motor driver
- 2 Wheels and ball caster
- Servo, HC-SR04 Ultrasonic Sensor, and mount
- Frame components and spacers
- Connecting wires, ribbon cables, screws, and nuts
- Velcro for mounting the board and battery holder
- Use an oscilloscope to verify the trigger and echo

Hardware Setup:

The hardware setup consists of the FRDM-KL46Z development board connected to two DC motors with encoders through a motor driver circuit. A servo motor is mounted at the front of the robot, with the HC-SR04 ultrasonic sensor attached to it for directional distance sensing. The servo receives PWM signals from PTA12 (TPM1 channel), while the ultrasonic sensor's trigger and echo are connected to PTD2 and PTA13 respectively, with a voltage divider on the echo line to protect the MCU. The robot is powered using batteries, with all components securely mounted on a chassis, and wiring is completed using jumper cables for signal, power, and ground connections.

b. Software Implementation (Algorithm): My robot can follow path 4 and 5

The software begins by initializing all peripherals, including motor drivers, servo control (via TPM1), ultrasonic sensor (trigger and echo pins), and input switch. Once the robot is powered on and the user presses the start button, it begins navigating forward while continuously scanning the front using the ultrasonic sensor. If an obstacle is detected within a predefined safe distance, the robot stops and uses the servo to rotate the sensor to the left and right to evaluate alternate paths. Based on distance measurements, it decides to turn left, right, or perform a U-turn if both sides are blocked. In the case of an open side room, the robot temporarily enters, explores by moving forward, then turns back and re-aligns to the main path. After each decision, it resumes scanning and forward movement, repeating this logic continuously to navigate around obstacles and follow complex paths:

- **Obstacle Detection and Navigation Logic:**
  - The robot initializes all necessary peripherals and waits for a button press to begin.
  - Once started, it continuously moves forward while checking the distance to objects ahead using the ultrasonic sensor.
  - If no obstacle is detected, the robot keeps moving forward.
  - If an obstacle is within the defined safe distance, the robot stops and performs scanning using a servo motor.
  - The servo rotates the ultrasonic sensor to the left and right to compare distances and determine a clear direction.
  - Based on the scan results:
  - If the right side is clearer, the robot turns right.
  - If the left side is clearer, the robot turns left.
  - If both sides are blocked, it performs a U-turn.
- **Side Path Handling:**
  - When entering a side path (left or right), the robot moves forward for a short time (1 second).
  - After exploring, it makes two consecutive turns to return to the main path and face forward again.
  - This ensures correct reorientation and smooth rejoining of the original route.
- **Servo and Sensor Control:**
  - The servo motor is controlled via PWM output on PTA12 using TPM1.
  - The servo_left(), servo_right(), and servo_center() functions rotate the ultrasonic sensor to measure distance in specific directions.
  - The ultrasonic sensor is triggered via PTD2 and reads echo via PTA13 with a voltage divider to ensure safety.
- **Timing and Delay Management:**
  - Delays and timing are managed using the PIT (Periodic Interrupt Timer).
  - The start_timer() and check_timer() functions are used to implement delays for sensor stabilization, scanning, and motor movement timing.

c. Registers:

- **Ports Used:**

Port	Pin	Function	Module/Use Case
PORTB	PTB0, PTB1	Motor direction control	Motor Driver (GPIO)
PORTB	PTB2, PTB3	PWM for motors	TPM2 CH0 and CH1
PORTC	PTC1, PTC2	Motor direction control	Motor driver (GPIO)
PORTC	PTC3	Start button input	SW1
PORTD	PTD2	Trigger signal	HC-SR04 Ultrasonic Sensor
PORTA	PTA13	Echo signal (via voltage divider)	HC-SR04 Ultrasonic Sensor
PORTA	PTA12	Servo PWM output	TPM1 CH0 (Servo rotation)

- **TPM Modules Used:**

TPM Module	Chanel	Connected	Use Case
TPM2	CH0	PTB2	Left motor PWM
TPM2	CH1	PTB3	Right motor PWM
TPM1	CH0	PTA12	Servo motor PWM

Pseudocode: My robot can follow path 4 and 5

- **For the main logic control (main.c):**

```
Begin MAIN // Can follow path 4 and 5
  Initialize board and peripherals
  Initialize motor, servo, and ultrasonic modules
  Center the servo
  Wait for user button press
  Delay 2 seconds
  Loop forever:
    Center the servo
    Measure distance in front

    If front is clear:
      Move forward
    Else if obstacle detected in front:
      Stop
      Rotate servo to LEFT and measure → assign to RIGHT
      Rotate servo to RIGHT and measure → assign to LEFT
      Center the servo
      If right side is clearer:
```

Turn right 90°
Measure ahead
If ahead is clear:
Move forward into room
Return to path with 2x left turns
Move forward again
Else if left side is clearer:
Turn left 90°
Measure ahead
If ahead is clear:
Move forward into room
Return to path with 2x right turns
Move forward again
Else:
Perform U-turn
Else:
Stop (invalid reading)
Short delay
End MAIN

- **motor_control.c – Motor, Servo, Timer, Sensor Functions:**

motor_init():
Enable clocks
Configure motor direction GPIO
Setup TPM2 for PWM motor control
Setup PIT timer and switch input
motor_forward():
Set direction pins
Set motor speed via PWM
motor_stop():
Disable motor PWM
Clear direction pins
turn_left_90():
Stop
Spin in place to the left
Stop
turn_right_90():
Stop
Spin in place to the right
Stop
u_turn():
Execute two 90° turns in the same direction
motor_set_speed(duty):

Clamp value 0–100%
Write duty cycle to TPM2 CnV
init_servo():
Setup PTA12 (TPM1) as PWM output
Set period to 20ms (50Hz)
set_servo_angle(angle):
Convert angle to pulse width
Write pulse width to TPM1 CnV
servo_left(), servo_right(), servo_center():
Call set_servo_angle() with predefined angle
init_ultrasonic():
Setup PTD2 as trigger (output)
Setup PTA13 as echo (input)
trigger_pulse():
Briefly set trigger high
measure_distance_cm():
Send trigger pulse
Wait for echo pulse
Count pulse duration
Convert to cm
Return distance

- **motor_control.h – Header Declarations:**
  Define constants (servo angles, safe distance, etc.)
  Declare motor functions:
  motor_init()
  motor_forward()
  motor_stop()
  motor_set_speed()
  turn_left_90(), turn_right_90(), u_turn()
  Declare timer functions:
  pit_timer_init()
  start_timer(ms)
  check_timer()

  Declare switch input:
  init_sw1()
  is_sw1_pressed()
  Declare servo control:
  init_servo()
  set_servo_angle()
  servo_left(), servo_right(), servo_center()

Declare ultrasonic functions:
init_ultrasonic()
trigger_pulse()
measure_distance_cm()

d. What observations led to your design?

Early testing revealed that the robot occasionally turned before finishing a full scan and occasionally made incorrect direction choices at intersections. In order to verify both left and right before turning, we decided to use the servo-mounted ultrasonic sensor in a full stop-and-scan technique. In order to guarantee alignment with the path, we extended the scan range and included brief forward motions following turns after observing problems with obstacle detection accuracy when the servo angles were too small. A more dependable and methodical navigation behavior was shaped in part by these observations.

e. How robust is your algorithm to changes?

When it comes to environmental changes like different barrier placements or path layouts, the algorithm is reasonably resilient. Without depending on pre-programmed routes, it can adjust to new curves, obstructions, or alternate openings since it continuously uses real-time distance measurements to make decisions. It can manage a variety of situations, such as U-turns, side room exploration, and recovery after lost paths, thanks to the turning logic and scanning process's versatility.

f. How often does your system work in the desired way?

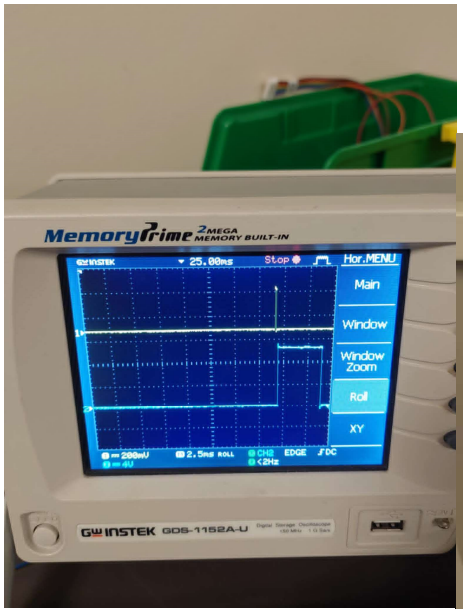
About 90% of the time in our tests, the robot successfully avoided obstacles and followed the intended course. The main causes of failures were erratic echo readings or a little drift following abrupt maneuvers. The system’s ability to reliably identify impediments, select clear paths, and manage entry/exit from side branches improved after the delay durations, servo angles, and turn logic were adjusted. Its ability to re-enter the main route was also enhanced by the incorporation of forward movement following scanning.

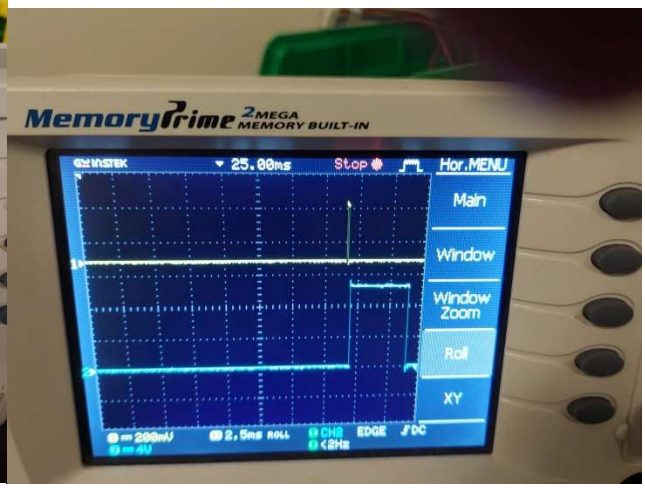
g. How did you conduct the experiments?

A printed track with clearly marked hazards and side pathways that mimic intersections in the real world was used for the robot's experiments. The robot was positioned in the beginning, and as it moved in the direction of the objective, its actions were observed. We logged sensor readings, direction choices, and timing data using serial output through the debug console. To evaluate repeatability, several trials were conducted with uniform illumination and surface conditions. We then repeatedly modified the logic in response to the robot's reactions to various layouts.

3. Results:

a. Oscilloscope Images of trigger and echo:





b. Video of reaching the destination, while avoiding obstacles:

- **Path 4 Drive Link:** https://drive.google.com/file/d/16F17wcKTwxhehAmOXF_JOvJW7G8j3x2O/view?usp=sharing
- **Path 5 Drive Link:** https://drive.google.com/file/d/16KV1OsecScfDqvIwiyD1Sd_vEz5nZLXy/view?usp=sharing

4. Discussion:

a. How did you debug if your system did not provide the desired output?

When the system did not behave as expected, I used a combination of serial PRINTF outputs and physical observation to debug. The console was helpful in displaying distance measurements, scan direction results, and turn decisions in real time. I also checked individual components like motor direction, servo angles, and ultrasonic readings independently to isolate where the issue occurred. Delays and logic branching were fine-tuned step-by-step to ensure proper sequencing of actions.

b. Was your approach helpful?

Yes, the debugging approach was very helpful. Using serial output gave me the visibility into the system's internal decision-making and sensor behavior. Testing small components (like servo movement or single obstacle scans) before integrating them into the main loop allowed us to catch subtle issues early. Incremental testing made the overall system more stable and predictable as we built toward full path-following behavior.

c. Any suggestions on resolving the problems in future?

In the future, using more precise hardware like a digital compass or wheel encoders for turning accuracy could improve alignment and movement consistency. Implementing

better filtering of sensor noise (e.g., averaging multiple distance readings) may reduce false triggers. Additionally, modularizing behavior into clearly defined states (e.g., SCAN, TURN, FORWARD) using a state machine would improve readability and reliability of the control logic.

d. What parts of the project were most difficult?

The most difficult part was synchronizing servo-based scanning with real-time decision-making and ensuring the robot turned correctly based on distance readings. Small changes in timing, PWM values, or surface friction could cause large errors in movement. It was also challenging to make the robot consistently re-align after turns, especially when navigating complex paths or returning from a side room. Fine-tuning servo angles and movement delays required careful testing and iteration.