CECS 347 Spring 2025

Hardware PWM Driven Romi Chassis Car

By

Eric Santana, Mason Doan, Hector Polanco, Kero Samaan

November 5, 2025

Romi chassis embedded system with Sharp sensor designed to follow objects and maneuver around walls. Uses hardware interrupts, analog to digital converter, pwm module, algorithms, and PLL. All programmed in keil using TM4C123 microcontroller.

Introduction

For project 2 we developed a working following robot using a romi car. The following robot consists of two modes: Mode 1 initiates an object follower and Mode 2 initiates a wall follower.

For **mode 1**, the robot follows an object at a distance of 20cm. If an object moves forward, the robot will move forward at a distance of 20cm. If an object moves backward, the robot will move backward at a distance of 20cm. When the object stops moving or if the object is out of range, the robot will stop. Additionally, if the object moves laterally (left or right), the robot follows accordingly, maintaining the same 20 cm distance offset.

For **mode 2**, the robot will be able to follow close to a wall depending which side of the robot is closer to the wall. It is important that the robot does not collide with the wall or any obstacles at any point of the wall follower function.

Onboard LEDs are used to determine the robot's current operational mode:

LED Color	Mode	Description
Red	Inactive	Robot does not move
Blue	Mode 1	Object follower
Green	Mode 2	Wall Follower

The system utilizes several key hardware components:

- **ADC1SS1**: Handles the analog-to-digital conversion of the three Sharp IR sensor outputs.
- **PLL (Phase-Locked Loop)**: Configures the system clock to operate at **16 MHz** for consistent timing and performance.
- **Port E**: Dedicated to IR sensor inputs.
- **Port F**: Controls the onboard LED indicators.
- **Port B**: Manages the motor control functions, including direction and hardware PWM (Pulse Width Modulation).

Our report explains the design, implementation, and testing methods used to create a reliable object and wall-following robot, along with possible improvements.

Development Steps & Task Assignments

Our project was completed in four steps part 1-4.

In **Part 1**, we tested the PWM hardware for the Romi car using example projects. We then modified the setup by assigning Port B to handle both the PWM hardware output and motor direction.

In **Part 2**, we tested the IR sensor outputs and collected readings from all three sensors at distances of 15 cm, 20 cm, 30 cm, and 70 cm to verify proper functionality. We noticed that the third sensor provided slightly inaccurate readings.

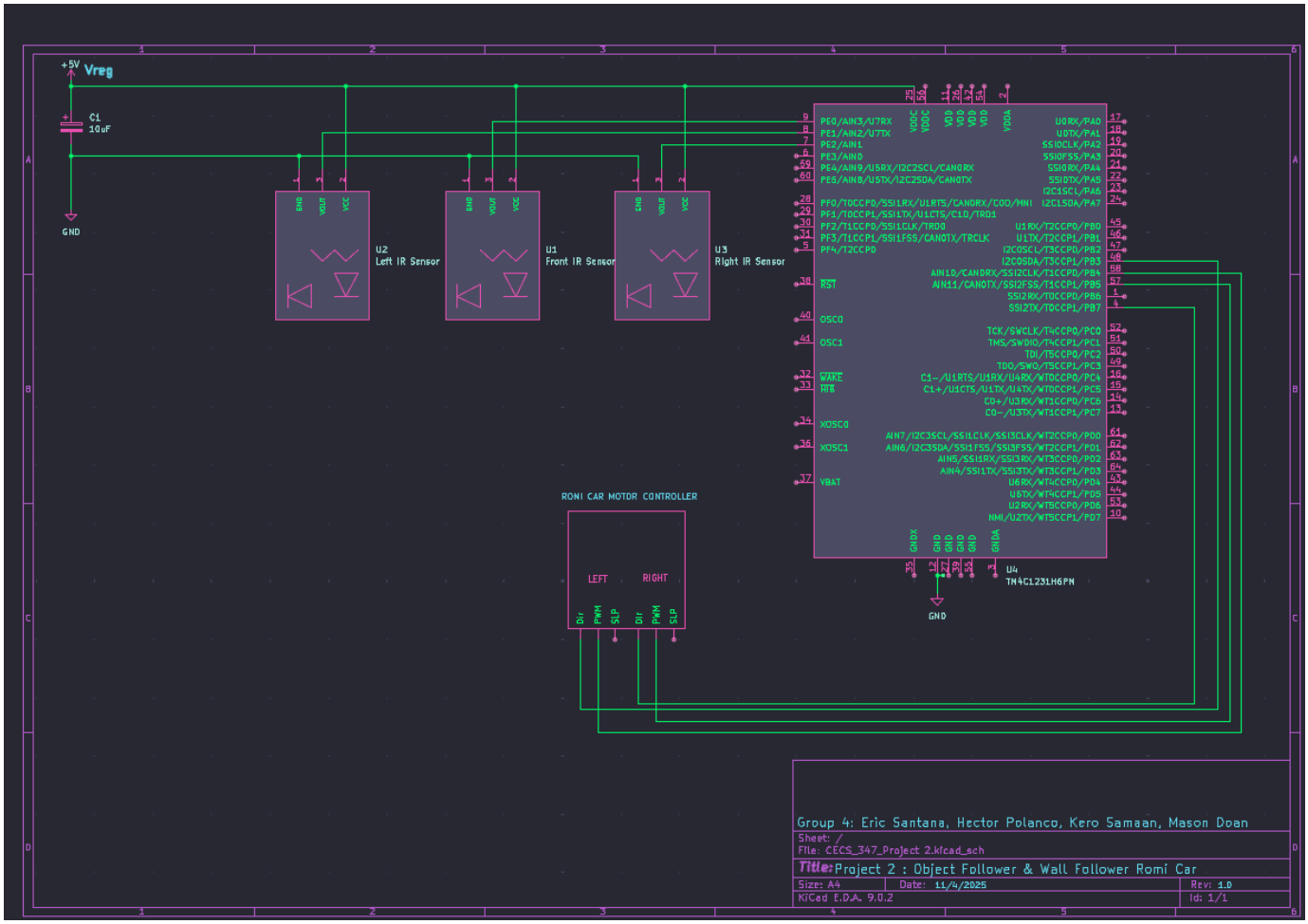
In **Part 3**, we developed the object-following function, which presented the most challenges. Implementing smooth turning behavior caused several issues with the forward and backwards following, and we also discovered hardware problems. The capacitor was incorrectly placed and was later corrected to connect between Vreg and GND. Additionally, the sensors were mounted improperly, which affected accuracy. After troubleshooting, we achieved semi-accurate sensor readings and improved overall system performance.

For part 4, we implemented control logic to change between the object following mode, and wall follower mode. The mode indicators consisted of a red led for inactive mode, blue led for object follower mode, and green led for wall follower mode. This step also implemented the wall follower function. This function also suffered from inaccurate sensor reading which affected the ability of the Romi car to properly execute the wall following function.

Team Member	Role/Responsibility	Tasks	Completion date
Hector Polanco	ADC / mode 1 and 2 implementation	Graph creation, HW/SW interaction description, code implementation	11/5/2025
Eric Santana	ADC / mode 1 implementation / PWM	Graph creation, development description, Hardware description, code implementation	11/5/2025
Kero Samaan	circuit connections/ debugging	Introduction and conclusion for report as well as hardware and software debugging	11/5/2025
Mason Doan	circuit connections/ debugging	Challenges and learning outcomes section of report as well as hardware and software debugging	11/5/2025

Hardware Design

Schematic


+5V
Vreg
C1
10uF
GND
U2
Left IR Sensor
U1
Front IR Sensor
U3
Right IR Sensor
RONI CAR MOTOR CONTROLLER
LEFT
RIGHT
Dir
PWM
SLP
U4
TM4C123LH6PM
Group 4: Eric Santana, Hector Polanco, Kero Samaan, Mason Doan
Sheet: /
File: CECS_347_Project 2.kicad_sch
Title: Project 2 : Object Follower & Wall Follower Romi Car
Size: A4
Date: 11/4/2025
Rev: 1.0
KiCad E.D.A. 9.0.2
Id: 1/1

The Romi Car system uses multiple components to implement the object following and wall following behavior. The main components are three infrared distance sensors, a TM4C123GH6PM LaunchPad microcontroller, a Romi Motor Controller board, and a 10 µF capacitor used for power supply stabilization.

The infrared distance sensors provide analog voltages that vary with the distance of an object in front of each sensor. Using three sensors positioned at the left, front, and right sides of the car, the system gains awareness of its surroundings to determine how close it is to an object or wall. Each sensor's output is connected to one of the analog input pins on Port E. These analog signals are read through ADC1 Sample Sequencer 1, then filtered using a median filter to remove noise and obtain stable readings. The sensors are powered by the 5 V regulator (Vreg) from the Romi Car battery pack, with a 10 µF capacitor connected between Vreg and GND to reduce voltage fluctuations and stabilize sensor performance.

The TM4C123 microcontroller serves as the main control unit, or "brain," of the system. It reads the processed sensor data and executes the control logic that determines the operating mode and movement behavior of the car. The microcontroller uses Port B for motor control. Specifically to generate PWM signals to control motor speed, and for motor direction signals.

The TM4C123 also controls the RGB LED indicators on Port F to show the current operating mode: red for inactive mode, blue for object follower mode, and green for wall follower mode. Together all these components work together to move the romi car in either object follower mode or wall follower mode.

Software Design

Modules:

PLL:

The PLL_Init function sets up the clock speed of the board to 16 MHz.

Port F:

When switch 1 or 2 is pressed, we enter the GPIOPortF handler, which is a falling-edge-triggered interrupt handler routine. When the ISR is entered, it will start by
Checking whether SW1 or SW2 was pressed. The robot starts in an inactive state, but when SW1 is pressed, the state variable mode will be assigned to enter the object_follower state by default. If SW2 is pressed while in the inactive state, the interrupt will have no effect. If SW2 is pressed while the robot is active, it will switch modes. In each case, the RIS register is first cleared, and then the mode variable is modified depending on the current status of the robot.

PWM:

The PWM_init function configures port B to be in charge of motor control using the integrated PWM module on the TM4C board. Pins B7 and B3 are configured for output with no alternative function, as they will be used to control the direction pins on the ROMI chassis. Pins B4 and B5 are configured for output as well, this time with alternative function set so that we can use the PWM module for these pins to control the speed of the motors. In our initialization function, the PWM module is set to go high when count = CMPA/CMPB.

There are many different functions that control how the motors will move, with them all directly setting the direction pins, but calling another function called duty, which controls the motor speed. The duty function takes two parameters to configure the duty cycle for two separate pins, B4 and B5. The duty cycle is the percentage of time that the voltage signal is high, so with our load at 15999, a 50% duty cycle will have the compare register PWM0_1_CMPA_R set to 7999.

ADC:

The ADC1_SS1_Init initializes port E since it has multiple pins that can be used as inputs for the analog-to-digital converter on the board. Pins PE0 (AIN3), PE1 (AIN2), and PE2 (AIN1) are configured for input in analog mode with alternative function to use the ADC module. Sample sequencer 1 is used for all three Sharp Sensors with the sample rate bits set to 0x00 for a rate of 125k samples per second.

ADC1_In321 is the module that actually takes the raw ADC input and assigns it to the parameters, which are pointer variables. This is done by first initiating sample sequencer 1 using the Processor Sample Sequence Initiate register. The three passed variables are then dereferenced and assigned the converted results for samples collected with the sample sequencer using the SSFIFO1_R register. The ICR register is then set to clear the RIS register to acknowledge completion.

The Median function takes three parameters to find which result out of the three is the median as a way to filter out any noisy results from ADC1_In321.

The ReadADCMedianFilter is what we use to get the actual values that decide which movement state the robot will be in. It has 3 parameters for each sensor,

which are pointer integers. 3 variables are then created for each sensor. The oldest variable keeps track of the oldest value from the ADC1_In321. The middle variable keeps track of the second-oldest value. The r variables are created so that their addresses can be passed to ADC1_In321 so that the pointer parameters can point to those addresses and give them ADC values. After each ADC sample, r is changed by ADC1_In321, and the oldest and middle variables are updated. These three values, r, oldest, and middle, are passed to the median function to find the median and assign that to the original parameters passed to ReadADCMedianFilter, which are what we use in our main program for controlling the robot.

HW/SW Interaction:

When powered on, the robot will start in an inactive state, which won't call any ADC or PWM functions. Once SW1 is pressed, it will trigger an interrupt to enter object follower mode, which will follow an object using the three interfaced Sharp Sensors. This is done by the PortF handler, which, when triggered, assigns the mode variable to a mode, either object follower or wall follower. The mode variable is used in a switch case so that when it is equal to OBJECT_FOLLOWER, it goes to the object follower case and calls the object follower function. It is the same with WALL_FOLLOWER, and the default case is to wait for an interrupt. Once the object follower is called, the ReadADCMedianFilter is used to take sensor inputs by taking raw inputs from ADC1_In321 and finding the median to get a more accurate reading. ADC1_In321 initiates the sample sequencer to take input voltages from the three pins PE0, PE1, and PE2. The voltages will then be converted to a digital value, which will determine how far the object is from the sensor. The closer the object is to the sensor, the stronger the infrared reflectance wave. This is how we can determine distances with different voltage values representing different distances. In our case, we measured that a distance of 10 cm, which is too close, will give a measurement of around 3000 from our sensor to the ADC, so this is used in our if statements to determine if the object the sensor is reading is too close and therefore must stop the chassis. Our algorithm works as follows: first, steer toward the target if it's stronger on one side. If the left sensor reads above the follow threshold and the right is below its threshold, the robot turns left. If the right sensor input is above the threshold, it turns right. The continue lines give side-steering priority, skipping the forward/back logic that follows.

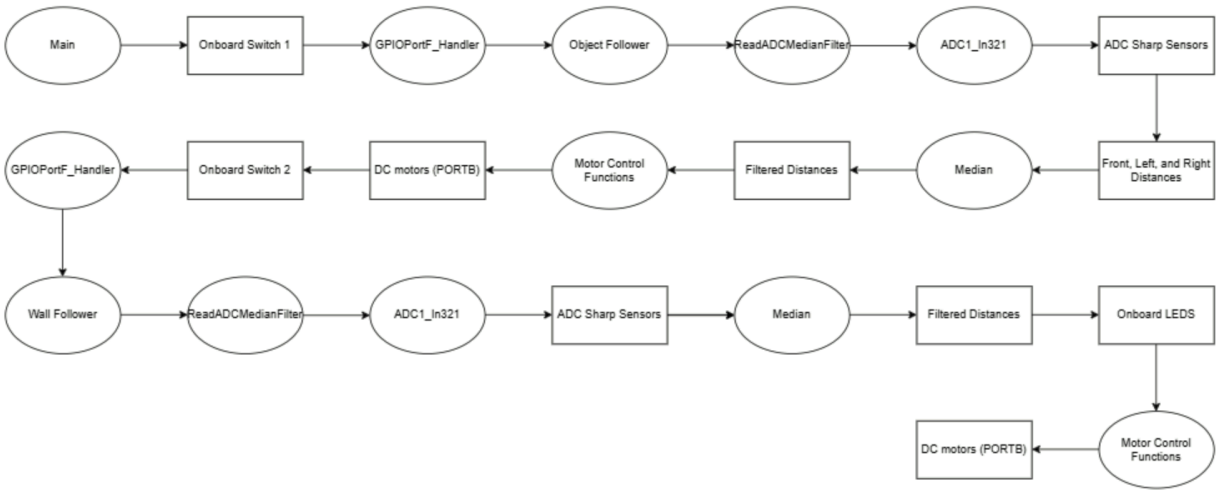
Otherwise, control distance using the front sensor. If the front reading is between MAXDIST and FRONT_FOLLOW, go forward to maintain following distance.

If the front reading is between FRONT_FOLLOW and FRONT_STOP (too close), go backward. If none of the above applies in the case of the sensors not sensing an object, stop. For the wall follower, the firstleft and firstright variables serve as a baseline (firstleft, firstright) to decide

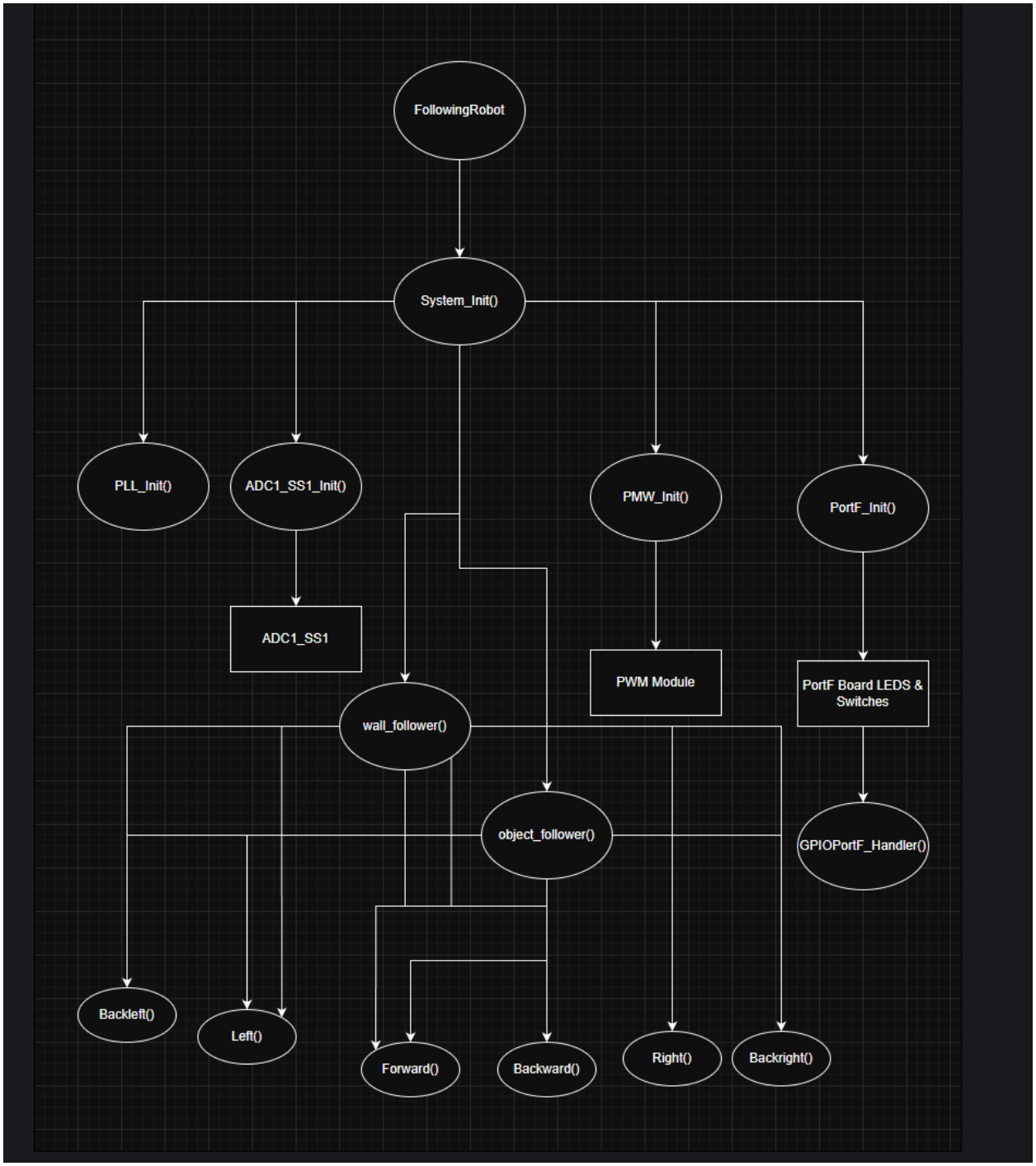
which wall the robot will follow. It enters WALL_FOLLOWER mode and continuously rereads the sensors. If the baseline indicates the left wall is closer than the right, the robot treats the left side as its reference: when the current left distance is at or beyond the wall threshold left >= WALL, it moves forwards along that wall, and in the case that it's not reading a wall or running into one, it turns left to realign to the wall. The same is the case with the right side. There is a comment for pivoting, which is when the front reading is too close and a side reading is also too close, so the car will pivot away from the wall.

The movement functions work by first setting the direction pins in software to whatever direction is appropriate for our desired action. By setting both of the direction pins to 1 using the mask 0x88, we set bits B7 and B3 to move both motors forward. Setting these pins to 0 will move that motor backwards. We can create turns moving forward by having both pins set to 1 and changing the duty cycle for the motors. The duty cycle determines the speed of the motor, so for a right turn, we will set the duty cycle for the inside wheel (right wheel) to be lower than the duty cycle for the outside wheel, thus making the car turn right. This is how we use software to control the motors. The hardware and software interactions for both modes are similar, with the only differences being the algorithms that determine which readings call for which movements and what movements are necessary for the modes. The actual software that controls the sensors and motors share identical implementations.

DATAFLOW CHART:

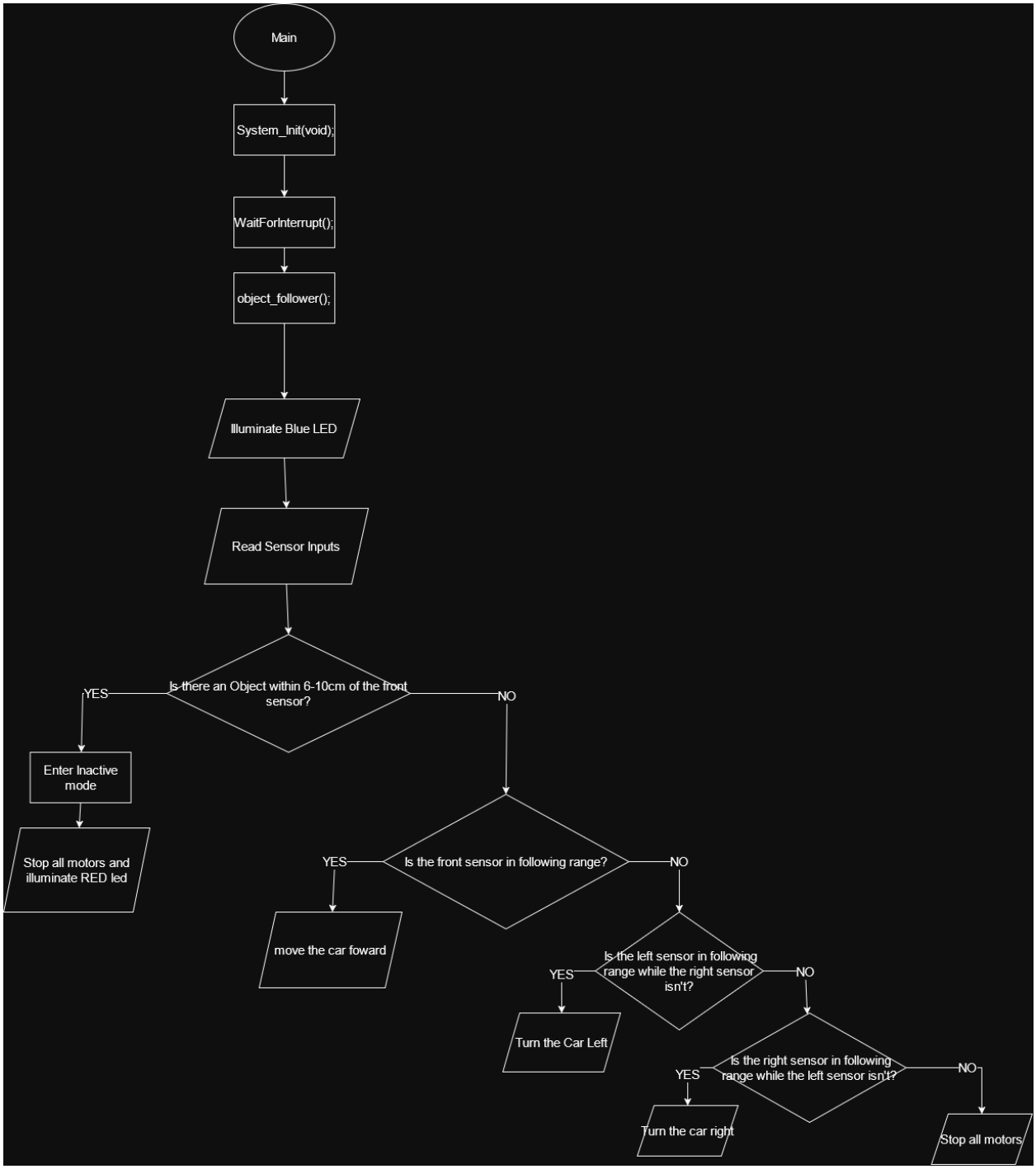


CALL GRAPH:

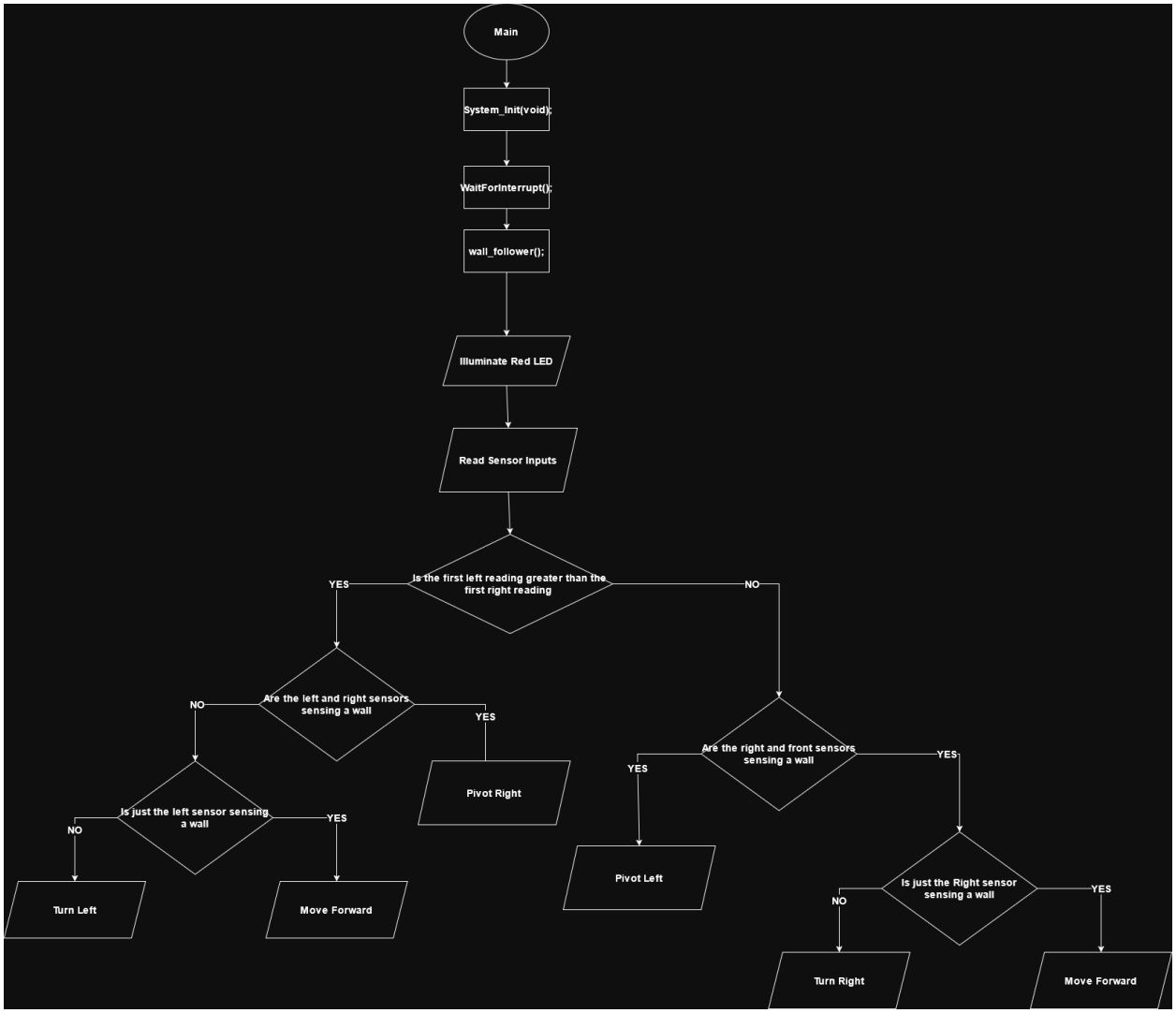

FollowingRobot
System_Init()
PLL_Init()
ADC1_SS1_Init()
PMW_Init()
PortF_Init()
ADC1_SS1
PWM Module
PortF Board LEDS & Switches
wall_follower()
object_follower()
GPIOPortF_Handler()
Backleft()
Left()
Forward()
Backward()
Right()
Backright()

FLOWCHART:

object:



Wall:



Conclusion:

Summary:

In this project we build a two-mode following robot using the TM4C123 launchpad and a romi-style car. We configured the PLL for a 16mhz clock, used hardware PWM on PB4/PB5 and direction pins on PB3/PB2 and PB7/PB6 to drive the DC motors, and read three IR sensors on PE1/PE0/PE2 using ADC 1 sample sequencer 1. Switch 1 activated/deactivated the robot, switch 2 selected between object

follower (mode 1) and wall follower (mode 2) and the RGB LEDs on PF1-3 showed the current status the robot is in.

challenges:

One of the biggest challenges was calibrating the IR sensor, since the ADC readings were nonlinear and sensitive to noise and surface changes. We had to test repeatedly to find the reasonable threshold for distances like 10 cm to 20 cm. Another big challenge balancing the motor duty cycles so the robot moved straight and turned smoothly, because small differences between the left and right motors caused the drift and inconsistent wall following. Also debugging switch interrupts and making sure the robot correctly entered and exited inactive states also took time, especially when we were testing modes back to back it made the testing and the troubleshooting much harder and more complicated. The greatest challenge was the fact that we couldn't figure out our sensor reading issues until the last day. As we were getting close to completing the project we seem to have messed up a board in a way thats hard to describe honestly. The board would do pivot movements even when the code for those movements were commented out and deleted and give false sensor readings. We were able to get video of our project somewhat working but because of the board issues we couldn't get a satisfactory result.

Learning Outcomes:

From this project we gained practical experience using GPIO, hardware PWM, ADC and timers together into one embedded system. We learned how to use the sensor reading to adjust the robot's speed and direction and how to implement and debug higher level behaviors like object following. Overall, this project strengthened our understanding of real time embedded systems and made us more confident working with small robotic cars.

References:

- **LECTURE SLIDES:** PWM.pptx, DC MOTORS.pptx, ADC and sharp IR sensor.pptx
- **TM4C123 Datasheet:** P. 354, P.1230 - 1236, p.847

- **Example projects:** Romi_Car_Test, Hardware PWM Car, ADCTestMain.c, FollowingRobot.c, FollowMe.c.