ECE 4437
Embedded Microcomputer Systems
Robot Project

Team: 18
Names: Christopher Andrew, Patrick Kelling

Project Description

The goal of this project was to produce a robot that can accomplish the following tasks.

- Navigate a maze via right turns and U turns at dead ends
- Use PID based control to avoid hitting walls
- Detect when different thicknesses of lines are crossed and perform different functions
    - Start and stop Transmitting PID Error data on thin line crossing via bluetooth and a modified modbus standard
    - Stop all function on thick line crossing
- System to add commands to test and control robot functionality

Project Demo's

Some of our recorded Demo's.

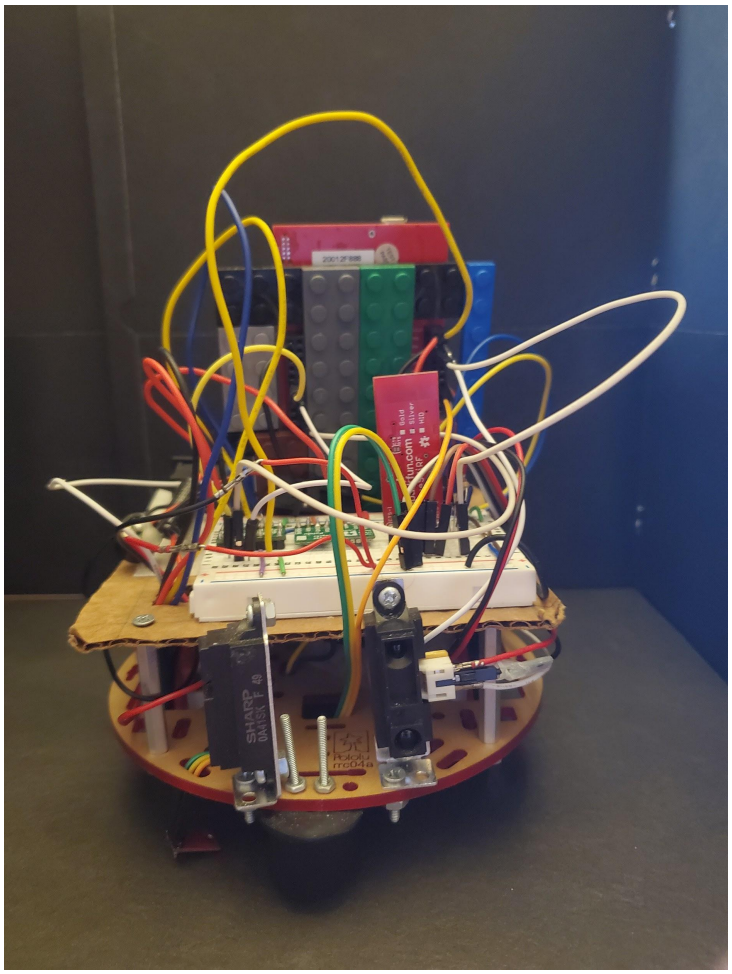
https://www.youtube.com/playlist?list=PLtCleJmhJJ_yT3hwMI-x0NeFqWM_-2Fyh

Hardware

Physical Layout

The construction of the robot was broken up into 3 different levels.

- Bottom layer
    - Motor and line sensing
- Middle layer
    - Battery pack and sensor mounts
- Top layer
    - Control Systems



Completed Robot

Bottom Layer

- Left Motor with wheel

- Right Motor with wheel
- 2 Coasters
- Light Sensor

Middle Layer

- Front Distance Sensor
- Right Distance Sensor
- Battery Pack

Top Layer

- Motor Controller
- 5 [V] Regulator
- 6 [V] Regulator
- Blue smirth Bluetooth module
- Tiva C
- Power Switch

Ports Used

Left GPIO Ports

- GND
- VBUS
    - Board power supply
- +3.3V
    - Vcc (Bluesmirth module)
    - Motor Controller
- PE4
    - TX-0 (Bluesmirth module)
- PE5
    - RX-I (Bluesmirth module)
- PB4
    - Side distance sensor Data
- PB5
    - Front DIstance Sensor
- PE1
    - Light sensor Data/Control pin
- PD0
    - Speaker Controller

Right GPIO Ports

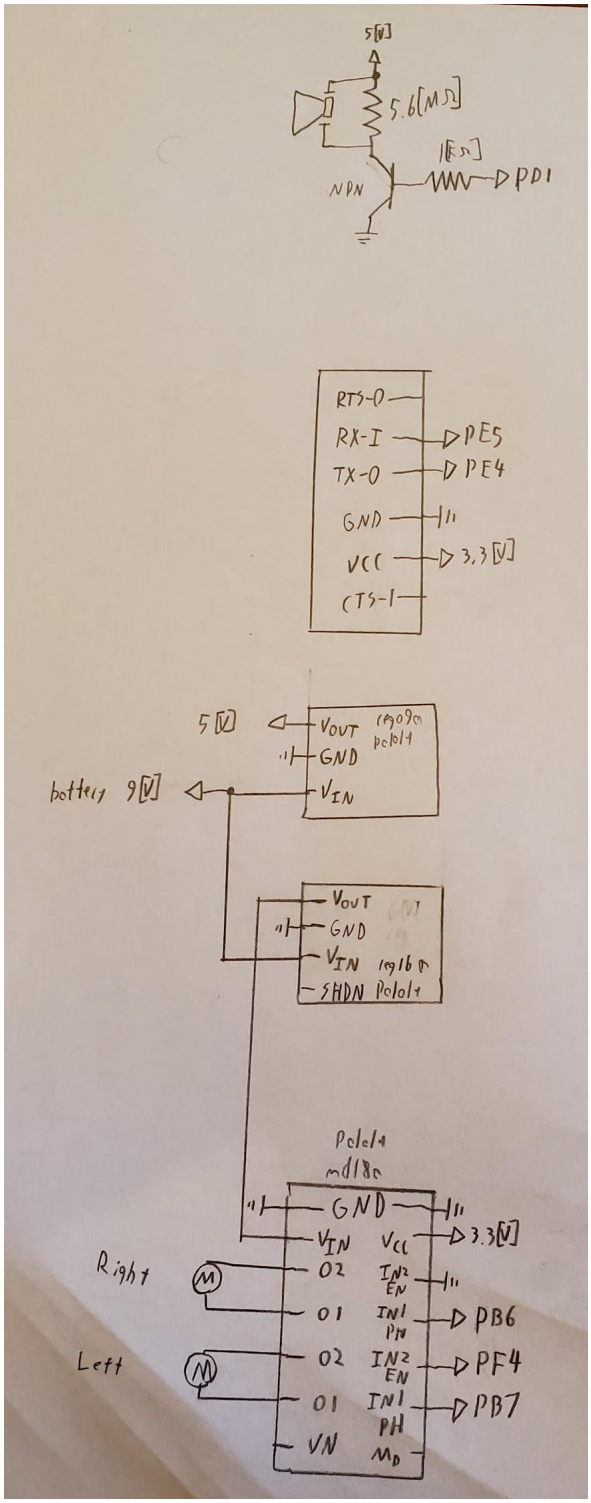
- GND

- PB6
    - IN1 PH Upper (Motor Driver)
- PB7
    - IN1 PH Lower (Motor Driver)
- PF4
    - IN2 EN Lower (Motor Driver)

Power Rails

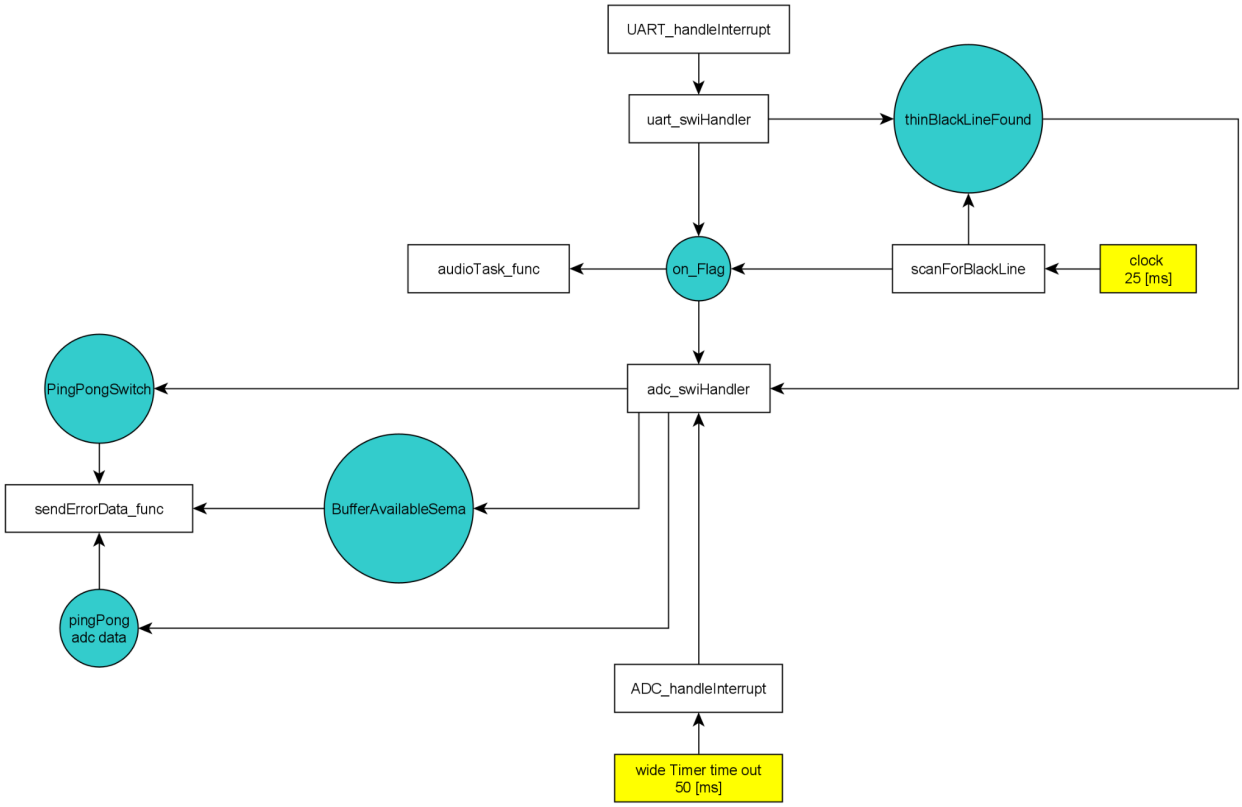
- 9 [V] (Battery supply)
    - 5 [V] Regulator (reg09a)
    - 6 [V] Regulator (reg16a)
- 6 [V] Regulator (reg16a)
    - Motor power VIN (md18a)
- 5 [V] Regulator (reg09a)
    - Forward/Right Distance Sensor
    - TivaC
    - Light sensor
- 3.3 [V] TivaC supplied
    - Motor Driver Logic Supply
    - Bluesmirth Power/Logic Supply

Circuit Diagram



Software

Outline of Software

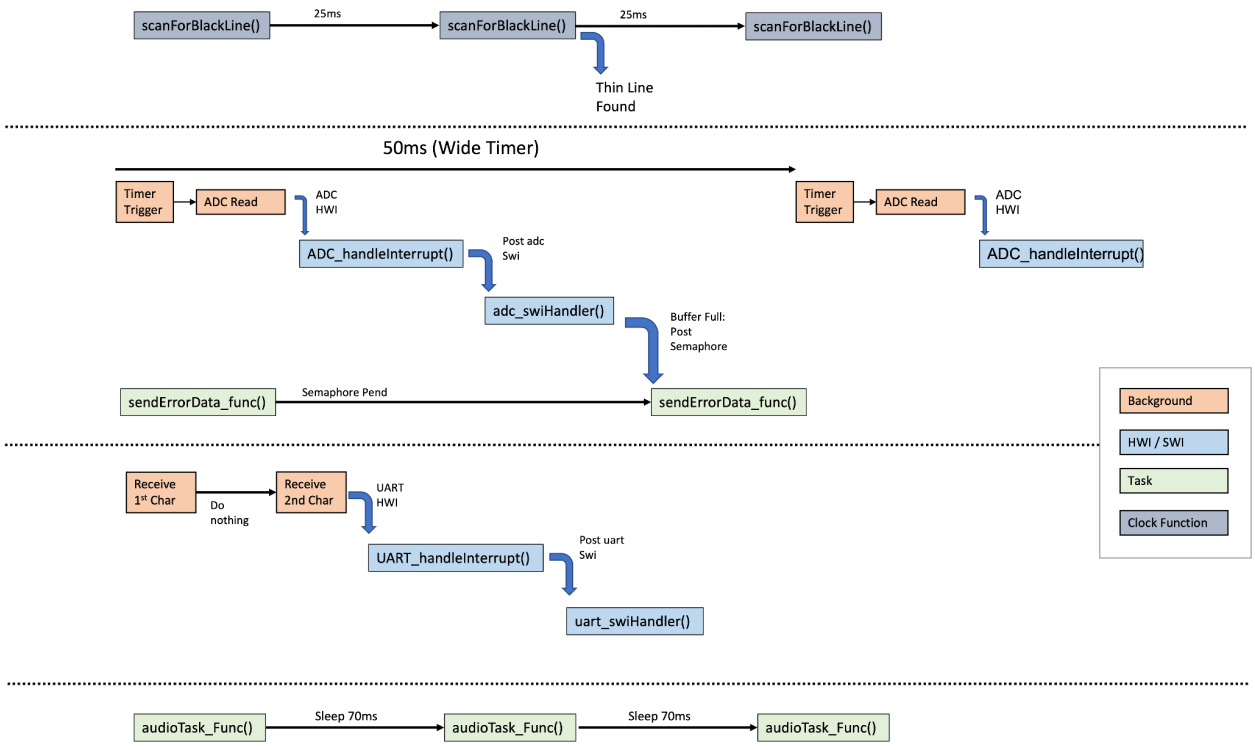


Our system has 7 threads:

- 2 HWI-SWI pairs
    - ADC Read (HWI) and PID/Motor Control (SWI)
        - ADC is triggered by a hardware timer every 50ms
        - When ADC read is finished, ADC HWI occurs
        - ADC values are saved and ADC SWI is posted, which handle motor control
    - UART Handling for Bluetooth Commands
        - Runs after 2 characters are received

        - Saves command and posts UART SWI to handle command
- Clock thread
    - Black Line Scanning
        - Runs every 25ms and reads light sensor
- 2 Task threads
    - Sending Error Data
        - Runs only when buffer is available (semaphore is posted)
    - Audio Output Handling
        - Lowest Priority Task, only runs in spare time
        - Controls PWM frequency to control pitch and play songs

Timing of Threads



Pseudo Code and Descriptions

Threads

ADC Read, PID, and Motor Control (HWI-SWI)

Purpose: This HWI-SWI pair handles the ADC reading and updating the PID every 50ms as well as handling turning arounds. It uses one of the Tiva C wide-timers to trigger the ADC read every 50ms. This is all done in the background and uses no CPU cycles. When the ADC read is finished, it triggers a HWI, which saves the ADC values and posts the ADC SWI to handle all motor control (PID and turnarounds) as well as data collection.

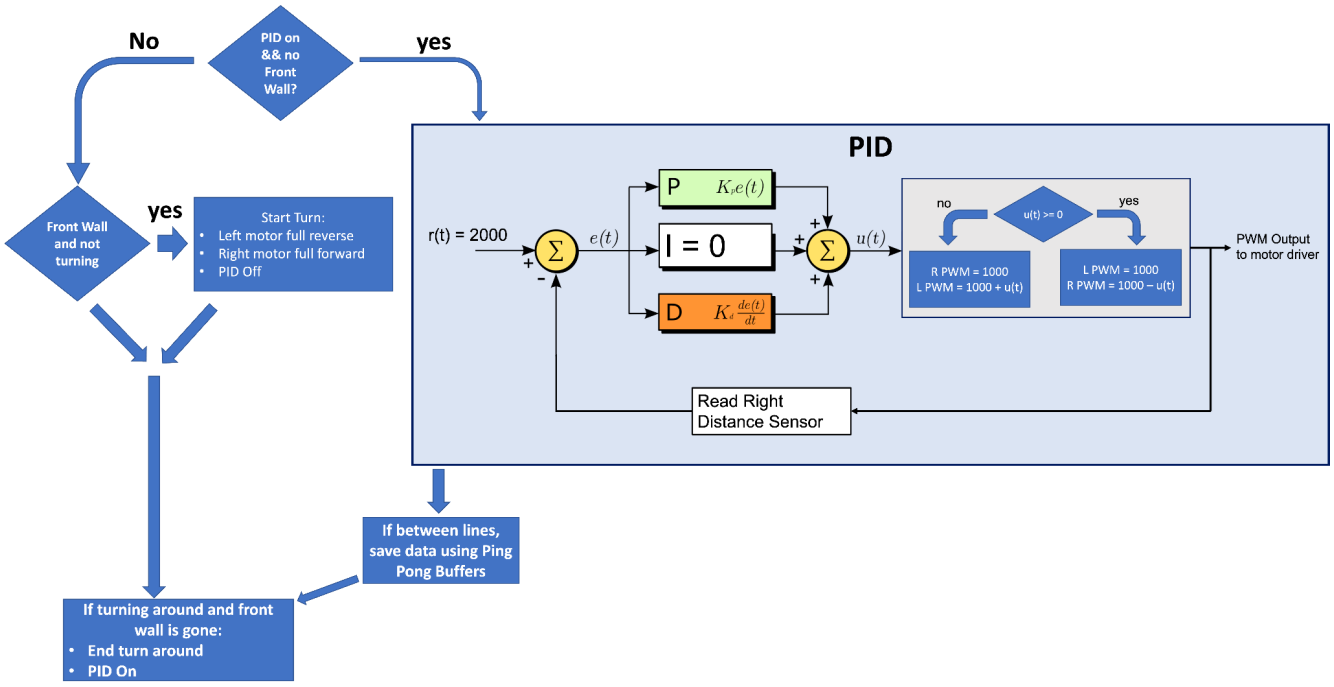
HWI - ADC_handleInterrupt() - Priority -1

1. Save ADC values to global variables
2. Post the ADC SWI handler

SWI - adc_swiHandler() - Priority -1

This block diagram of SWI is shown below. This will only run if the robot is on (On_Flag is set). The PID will only run if the robot is not too close to the front wall, otherwise the SWI executes a turn around and stops when the front wall is no longer close.

The PID works by controlling the PWM output difference between the two motors. Each motor PWM has a period of 1000 clock cycles, and the PID determines the difference in ticks per cycle between the motors. One of the motors is always at full power (100% duty cycle) to maximize the speed.



If the thinLineFound flag is raised the raw adc value is written to the ping buffer every other call of the adc_swiHandler(). When 20 items are stored in the buffer we switch to storing information in the pong buffer and post a semaphore singling sendErrorData_func to run and transmit the data. In addition while the thinLineFound flag is raised the blue board LED will be turned on showing that data is being collected.

UART Handling (HWI-SWI)

Purpose: This HWI-SWI pair of threads handles the bluetooth commands through a UART receive pin. The HWI is triggered when the UART FIFO has received 2 character inputs and the SWI handles the command interpretation.

HWI - UART_handleInterrupt() - Priority -1

1. Save both command characters
2. Post the SWI UART handler

SWI - uart_swiHandler() - Priority -1

1. Output that the command was received and what command it is.
2. Call the function associated with that command in the command lookup table using a command lookup function and function pointers
    a. If the command exists, the associated function is called
    b. If the command doesn't exist, the last function in the lookup table is called: InvalidCommand() which outputs that the command was invalid

Command Table:

Command	Description	Associated Function
GT	Toggle Green LED	GreenLEDToggle()
BT	Toggle Blue LED	BlueLEDToggle()
RT	Toggle Red LED	RedLEDToggle()
FR	Read Front ADC	ReadFrontADCValue()
SR	Read Side ADC	ReadSideADCValue()
RO	Right Motor On	motorRStart()
LO	Left Motor On	motorLStart()

RH	Right Motor Off	motorRStop()
LH	Left Motor Off	motorLStop()
RF	Right Motor Faster	motorRF()
RS	Right Motor Slower	motorRS()
LF	Left Motor Faster	motorLF()
LS	Left Motor Slower	motorLS()
OF	Turn off Robot (unset On_Flag)	PIDOn()
ON	Turn off Robot (set On_Flag)	PIDOff()
CE	Data Collection Enabled	collectDataEnable()
CD	Data Collection Disabled	collectDataDisable()
Anything Else	Invalid Command	InvalidCommand()

Black Line Scanning (CLOCK function)

Purpose: This thread's goal is to figure out what color of surface the light sensor is over and then to raise or lower different flags depending on how long the sensor is over dark area. This function is fired every 25 [ms] by the RTOS clock

scanForBlackLine()

1. Set PE1 to be a digital output pin
2. Write 1 to PE1
3. Delay 0.16 [ms] to allow sensor capacitor to charge
4. Get a start timestamp from Timestamp_get32()
5. Set PE1 to digital input pin
6. Wait while reading the input till the pin goes low
7. Capture the end timestamp
8. Calculate the decayTime by subtracting (endTime - start)/80 to get delay time in [us]
9. If decayTime > 3500 [us]
    a. A black line was found so we increment blackCount
10. If decayTime is not > 3500 [us] we moved off the black so its time for the line type handling
    a. If blackCount >= blackThreshold a thick black line was found
        i. Turn off PID
        ii. Turn off Blue LED

iii. Set thinLineFound to 0
iv. Send to bluetooth “\r\n”
v. Send the robot run time in 10th’s of a second
vi. Send to bluetooth “s\r\n”

b. Else if blackCount >=1 && blackCount < blackThreshold thin line found
   i. Toggle thinLineFound ^= 1
   ii. Toggle blue LED
   iii. If thinLineFound == 0
      1. PingPongSwitch is toggled
      2. Post BufferAvailableSema
c. blackCount = 0 as a non blackLine was found

Audio Output Task (Lowest Priority Task)

Purpose: This task's control a PWM frequency to output music from a Piezo buzzer. The pitch is determined by the frequency, and the PWM is always operated at a 50% duty cycle to produce a square wave.

The song is saved in “Songs.h” as an array of notes followed by each note’s length as well as the songs array length.

audioTask_Func(UArg arg0, UArg arg1) - Priority 1

1. Start of function
   a. If Song is playing and current note has ended (number of 16th == 0)
      i. Task sleep 8ms to create space between notes
      ii. Get Frequency of next note from array
      iii. Get the length of the next note from the array (8=8th note, 4=quarter note, etc...)
      iv. Convert the length to the number of 16th’s of a measure to play note and save this value globally
      v. Set PWM tick period based on the note so frequency is correct
      vi. Set PWM duty cycle to 50% by setting number of ticks on to half the period
   b. Else If in the middle of a note
      i. Subtract a 16th from the number of 16ths to play note
   c. Else Jump to next song if one is available
2. Sleep 50ms

Error Data Output Task (Task)

Purpose: This task's goal is to transmit collected data over bluetooth and a modified form of the mod bus standard when a semaphore is posted. In addition the built in green LED is turned on during data transmission.

sendErrorData_func(UArg arg0, UArg arg1) - Priority 2

1. Wait for semaphore to start transmission
2. Turn on green LED (Generally not on long enough to see)
3. Check if fineLineFound == 0
    a. If it is found this is the last run so this data transmission will e changed to not be the full length
4. Output via UARTcharPut the header ":18"
5. For every data point in the buffer
    a. Select the buffer not actively being written to buffer via ping pong system and extract a data point
    b. Convert to [cm] and subtract from midpoint
    c. Take the absolute value of the converted data
    d. Use sprintf to convert data to hex string
    e. Output this single hex character
    f. Repeat until the buffer is empty
6. Print footer "18\r\n" via UARTCharPut
7. Turn LED off
8. Go to step 1

Miscellaneous Functions

main()

Purpose: This function performs initial setup of the robot before handing control off to the RTOS.

1. Run hardware initialization.
2. Turn off the PID and thin line found
3. Start bios

Void Board_Init()

Purpose: initializes all needed hardware peripherals and GPIO Pins.

Int findCommandNum(char str[2])

Purpose: searches the command_loopup table for the provided string and returns the command number if it is found.

1. For i <number of commands
2. If str == command
    a. Return i
3. End for
4. Return i

Void GreenLEDToggle()

Purpose: toggles the state of the Green LED on the tivaC

1. Read PF1|PF2|PF3 status
2. Invert PF3 save to pinstatus
3. Write pinstatus to port F

Void BlueLEDToggle()

Purpose: toggles the state of the BlueLED on the tivaC

1. Read PF1|PF2|PF3 status
2. Invert PF2 save to pinstatus
3. Write pinstatus to port F

Void RedLEDToggle()

Purpose: toggles the state of the Red LED on the tivaC

1. Read PF1|PF2|PF3 status
2. Invert PF1 save to pinstatus
3. Write pinstatus to port F

ReadFrontADCValue

Purpose: Print out last read Front ADC value as float of distance in cm

1. Convert ADC value to a float of the distance
2. Convert that float to a character array
3. Output the character array one at a time

ReadSideADCValue

Purpose: Print out last read Side ADC value

1. Convert ADC value to a character array
2. Output that character array one character at a time

motorRStart

Purpose: Turns the right motor on and sets it to full speed

1. PWMPulseWidthSet(PWM0_BASE, PWM_OUT_0, 1000); //PB6

motorLStart

Purpose: Turns the left motor on and sets it to full speed

1. PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1, 1000); // PB7

motorRStop

Purpose: Turns off the right motor

1. PWMPulseWidthSet(PWM0_BASE, PWM_OUT_0, 1); //PB6
2. NOTE: PWMPulseWidth can not be set to 0

motorLStop

Purpose: Turns off the left motor

1. PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1, 1); //PB7

motorRF

Purpose: Makes the right motor faster

1. if(PWMPulseWidthGet PWM0 PWMO0 <= 900
    a. Set pulse width to current pulse width + 100
2. Else
    a. Print to bluetooth via UARTCharPut
    b. MaxSpeedRightReached\r\n

motorRS

Purpose: Makes the right motor slower

1. If PWMPulseWidth PWM0 PWMO0 >= 100
    a. Set pwm pulse width to be 100 less than current pulse width
2. ELSE
    a. Print to bluetooth via UartCharPut
    b. MinSpeedRightReached\r\n

motorLF

Purpose: Makes the left motor faster

1. if(PWMPulseWidthGet PWM0 PWMO1 <= 900
    a. Set pulse width to current pulse width + 100
3. Else
    a. Print to bluetooth via UARTCharPut
    b. MaxSpeedRightReached\r\n

motorLS

Purpose: Makes the Left motor slower

1. If PWMPulseWidth PWM0 PWMO1 >= 100
    a. Set pwm pulse width to be 100 less than current pulse width
2. ELSE
    a. Print to bluetooth via UartCharPut
    b. MinSpeedRightReached\r\n

PIDOn

Purpose: Sets On_Flag to start the robot.

1. On_Flag = 1

PIDOff

Purpose: Unsets On_Flag to start the robot.

1. On_Flag = 0

InvalidCommand

Purpose: Prints out that the command was invalid

1. Print out Invalid Command comment via uart to bluetooth

delay(uint32_t i)

Purpose: Uses the system delay function to delay a certain amount of time (16ms * i).

1. Call System delay function with parameter of (13400 * i)

GreenLEDOn

Purpose: Turns on onboard green LED on Tiva C.

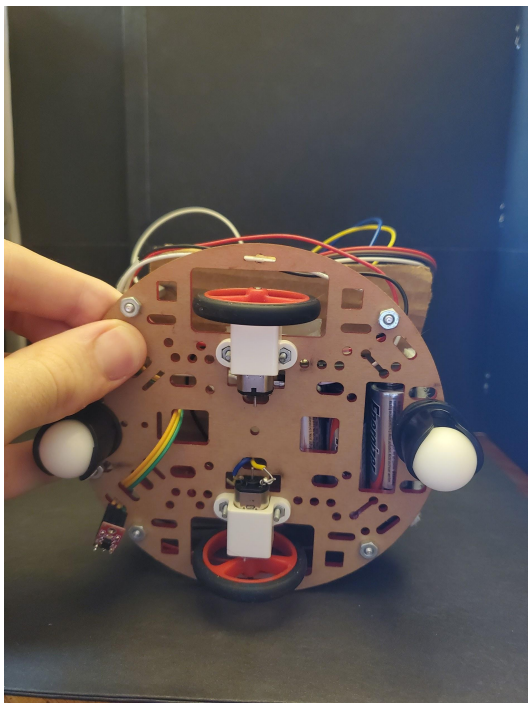
1. Turn on green LED pin

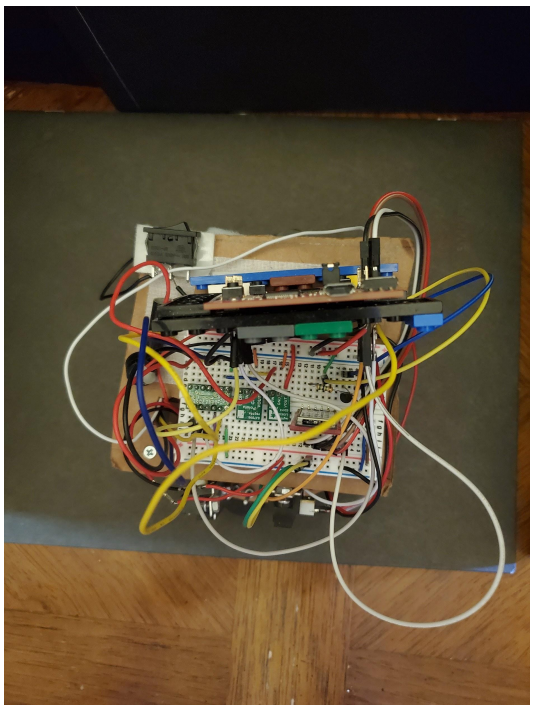
GreenLEDOff

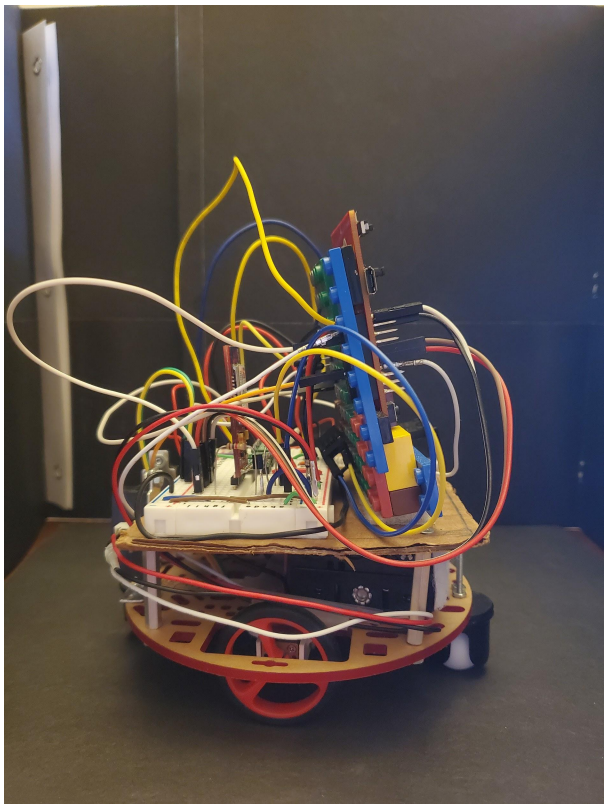
Purpose: Turns off onboard green LED on Tiva C.

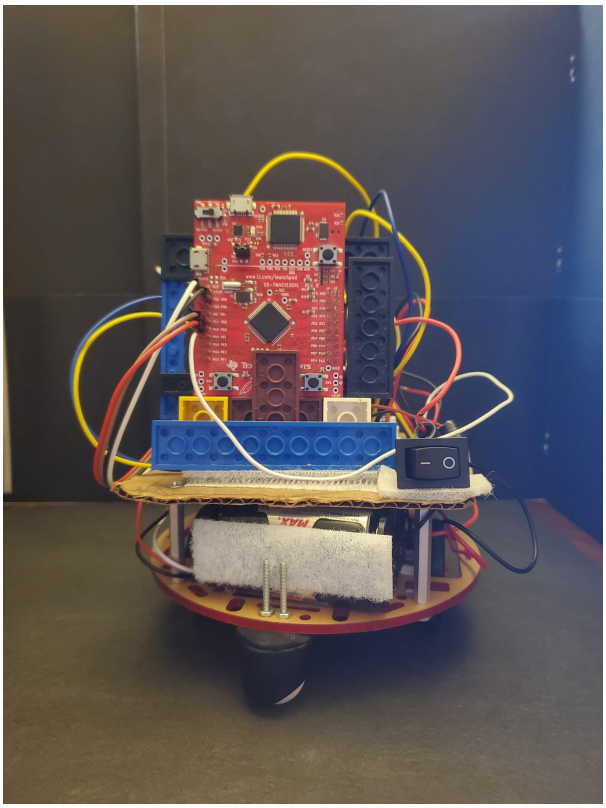
1. Turn off green LED pin

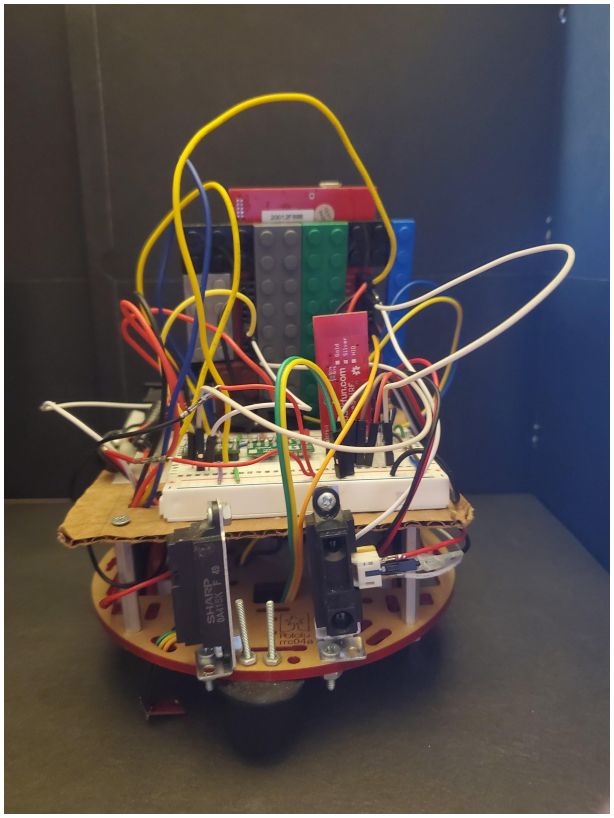
Appendix: Robot Images










SHARP