Department of Electrical Engineering University of Washington

E E 474 Lab 4

Interrupts and Realtime

The report and its contents are solely our team's work and cites outside references for work that is not our own.

Signature:
Xiaoyi Ling, 1428502
Tiffany Luu, 1435225
Pezhman Khorasani, 1640743

TABLE OF CONTENTS

Introduction

In this lab, we learned to program and implement an H-bridge driver to control both motors, and used these motors run the tank. We also learned used distance sensors to observe obstacles, by reading the output of these sensors through ADC pins. We also practised doing timer interrupts by using the signal() function in C. When something is too close to the sensor, the sensor program sends a signal to the master program. In this case, the moving process of the tank will be interrupted when one of the sensor find obstacles on either side and the tank will move in a different direction correspondingly.

Design Specification

Tank Autodrive Mode *(drive.c, sense.c)*

Input: Distance from obstacles
Output: The tank will automatically drive and turn in different directions
Error check: The program is able to check if a file is opened successfully. If not, the system will print error message to the terminal and exit

This program will run the tank when started and automatically drive to avoid obstacles

- When the Beaglebone is powered on (via the power bank), the script to run the programs will run on boot up.
- The tank's default is to move forwards, but it will not run until the switch is turned on.
- Four distance sensors are attached to the four sides of the tank to sense the distance to any object from each direction. When there are obstacles on one side of the tank, the corresponding sensor will sense and the tank will turn in another direction.
- If there are obstacles in the front of the tank, the tank will reverse; if there are obstacles on the back of the tank, the tank will move forward. (Not shown in demo due to motor/wheel issues: if there are obstacles on the left, the tank will turn to right; if there are obstacles on the right, the tank will turn to left.)
- If the switch is turned off, the tank will stop again.

H-bridge to Drive the Tank *(motor_test.c)*

Input: N/A
Output: H-bridge controlling the moving wheels of the tank
Error check: The program is able to check if a file is opened successfully. If not, the system will print error messages to the terminal and exit.

This program will control the h-bridge driver that moves the wheels of the tank

- Both wheels with move forward when the program starts.
- After 3 seconds, both wheels will move backward to back the tank.
- After 3 seconds, left wheel will move forward and right wheel will stop so the tank will turn right.

- After 3 seconds, left wheel will stop and right wheel will move forward so the tank will turn left.

Sensor Inputs *(ds.c)*

Input: Distance sensors
Output: Terminal window
Error check: The program is able to check if a file is opened successfully. If not, the system will print error message to the terminal and exit

This program will read the input from all four distance sensors and print the value to the terminal window.

- The ADC reading is an input voltage from 0-1800 mV.
- The max reading from the distance sensor will be 1600.
- The default is set to 1000 total readings, each sampled per 0.5ms

ADC Sampling with Software Timer Interrupt *(sender.c, receiver.c)*

Input: Distance sensors
Output: Terminal window
Error check: The program is able to check if a file is opened successfully. If not, the system will print error message to the terminal and exit

The receiver program should be run before the sender program. Bash scripts will compile and run both programs with the correct arguments.

- Four sensors are attached to each side of the tank. The are designated as “front, back, right, left” sensors.
- When an obstacle comes within ~12 cm of the sensor, the sender program will send a signal to the receiver program. It will print this to its terminal (example: “Signal for front sent”).
- When the sender receives the signal, it will distinguish which sensor sent the signal and print this to its terminal (example: “received front”).

Hardware

The following pieces of hardware were used in this project.

H-bridge Motor Driver

An H-bridge motor driver (model TB6612FNG) was used to control the motors attached to the wheels of the tank. Input/output A was connected to the left motor, while input/output B was connected to the right motor. The PWM for each motor is used to control the speed the motor turns. The smaller the period, the faster the motor turns. The wheels will turn in either direction depending on whether IN1 or IN2 is high. The connections for the H-bridge are shown in Figure 1 and Table 1 below.

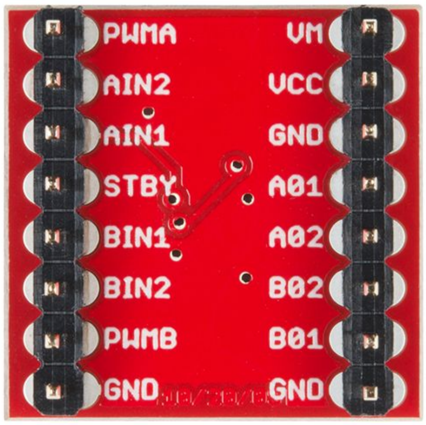


Figure 1: Image of the pin connections for the h-bridge driver

Table 1: H-bridge Pin Connections

H-bridge	Connections	Function
PWMA	EHRPWM1A	chA PWM input, controls the speed of motor A
AIN2	GPIO_66	chA input2, instruction pin for motor A
AIN1	GPIO_69	chA input1, instruction pin for motor A
STBY	Switch	No output to the motor when standby is low
BIN1	GPIO_47	chB input1, instruction pin for motor B
BIN2	GPIO_27	chB input2, instruction pin for motor B
PWMB	EHRPWM2A	chB PWM input, controls the speed of motor B
GND	GND	Power: GND (0 V)
VM	Battery pack	Power: ~6.5 V, power for the motor

VCC	SYS 5V	Power: 5 V, power for the H-bridge
GND	GND	Power: GND (0 V)
A01	(red) Left Motor	chA output1, output to motor A
A02	(black) Left Motor	chA output2, output to motor A
B02	(black) Right Motor	chB output1, output to motor B
B01	(red) Right Motor	chB output2, output to motor B
GND	GND	Power: GND (0 V)

For the standby pin, we connected an SPDT switch where the middle was connected to STBY, and the other two pins were connected to 0 V and 3.3 V. The logic table for the input pins to control the motor movement is show below in Table 2. The forward/reverse can be interchangeable depending on how the output is hooked up to the motor.

Table 2: H-bridge Controls

IN1	**IN2**	**Behavior**
0	0	Free
0	1	Forward
1	0	Reverse
1	1	Brake

Distance Sensors

We connected four distance sensors to each side of the tank. An image of this is shown below in Figure 2.

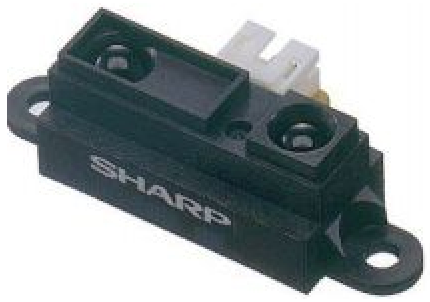
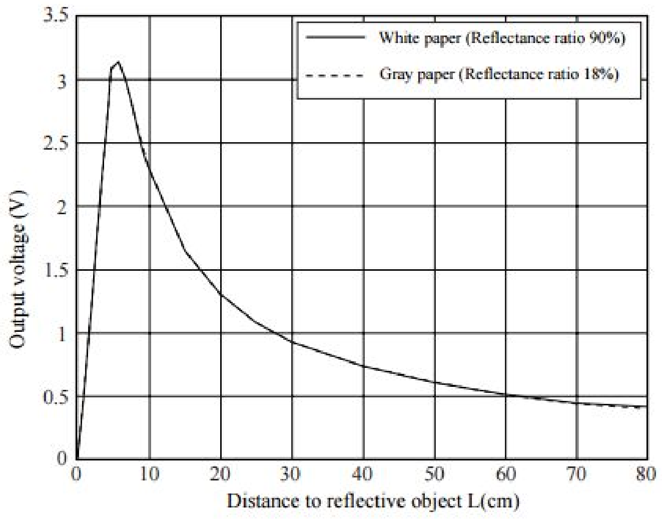


Figure 2: Distance sensor.

Because the max output of the distance sensor is greater than the minimum input of the ADC pins of the Beaglebone, we used a voltage divider to divide the output in half.



We used two 10kΩ resistors to form a voltage divider for the sensor since the maximum input to the AIN pins is 1.8V. We use 0.9V input to the AIN pins to be the value when the tank should change direction. Considering the voltage divider, the actual output of the sensor should be 1.8V. From the data on the left, we can see that the approximate distance is about 15 cm.

Figure 3: Datasheet for distance sensor

Thus the three pins of the distance sensor were connected to the following connection described in Table 3 below.

Table 3: Connections of the Distance Sensor

Connect to	Connections	Function
Beaglebone GND	GND	Ground
Beaglebone SYS 5V	Vcc	Power: 5 V
Voltage dividers	Vout	Output voltage corresponding to the distance to the obstacles in front of the sensor.

Most of the power for the entire system and the inputs were provided by the Beaglebone. Table 4 describes how the Beaglebone pins were connected to the H-bridge and sensors as inputs/outputs. Note that in addition to the sensor inputs, we blinked an LED every time one of the sensors detected an obstacle.

Table 4: Beaglebone Connections

Beaglebone	Connections	Function
SYS 5V	Breadboard +	Power: V_{CC} = 5 V
DGND	Breadboard -	Power: V_{DD} = 0 V
GPIO_44	LED forward	Light up LED when signal from front sensor is received

GPIO_68	LED back	Light up LED when signal from back sensor is received
GPIO_67	LED right	Light up LED when signal from right sensor is received
GPIO_26	LED left	Light up LED when signal from back sensor is received
VDD 3.3 V/ DGND	Switch	Control the entire tank to move when the switch is on and stop when the switch is off
AIN0	Voltage divider for front sensor	Read the input voltage from front sensor after it is divided to half by a voltage divider
AIN1	Voltage divider for back sensor	Read the input voltage from back sensor after it is divided to half by a voltage divider
AIN2	Voltage divider for right sensor	Read the input voltage from right sensor after it is divided to half by a voltage divider
AIN3	Voltage divider for left sensor	Read the input voltage from left sensor after it is divided to half by a voltage divider

Software

The following is a short description of the C files and programs that are used to move the tank, to send signals from the distance sensors to the drive program, and to receive signals and interrupt current process to turn the direction of the tank.

Tank Autodrive Mode

startup.sh: Script that runs when the Beaglebone is booted up. It will run the drive program, find the process ID of the drive program, and then run the sense program with that process ID as one of the arguments.

start.sh: Performs the same tasks as startup.sh, but not on boot up. It also compiles the code for sense.c and drive.c before running the programs and should be run in the folder where sense.c and drive.c are.

sense.c: Reads input from four AIN pins of Beaglebone that are connected to the distance sensors every 100us. It computes the average value every 100 samples. It will then check the average value in the order of front, back, left and right. If any value is larger than 900, we will send the corresponding signal to interrupt the drive program. In our case, the signals are 20 for front, 30 for back, 40 for left and 50 for right, sent to the process ID of drive which should be an argument input into the program.

drive.c: It will drive the car according to the signals received from the sensors. The tank will initially move forward. When signals are received, current process will be interrupted and the tank will move to another directions according to the signal. When signal 20 is received, it implies that there are obstacles in the front and the tank will move backwards; when signal 30 is received, it implies that there are obstacles in the back and the tank will move forwards. (In program, but not shown in demo: when signal 40 is received, it implies that there are obstacles on the left and the tank will move to right;when signal 50 is received, it implies that there are obstacles on the right and the tank will move to left).

H-bridge to Drive the Tank

motor_test.c: This program opens system files for motor driver and make wheels move forwards and backwards by setting the values for gpio pin files that corresponds to the pins of motor driver. It will make the tank to move forward for 3 seconds, move backward for 3 seconds, only move left wheels for 3 seconds, only move right wheels for 3 seconds, and then brake.

Sensor Inputs

ds.c: This program reads inputs from the distance sensors through the ADC pins. It reads an input value in milliVolts between 0-1800. It will then print these values to the terminal window. This process repeats 1000 times every 0.5 ms.

ADC Sampling with Software Timer Interrupt

receive.sh: Script that compiles and runs the receiver.c program

receiver.c: When the distance sensors sense something close to them, a signal will be sent to this program. This program will print which sensor signal it received to the terminal window.

send.sh: Script that compiles and runs the sender.c program, giving the process ID of the receive process as an input.

sender.c: Reads input from the ADC pins of the Beaglebone. These are connected to the distance sensors on each side of the tank. It samples the pins per 1ms, and the average value of these samples per 100 samples is taken. If this average exceeds the threshold value of 1000 mV, then a signal will be sent to interrupt the receiver program. The program will print which signal was sent to the terminal window.

Testing

Motors/H-bridge

Difficulties we met during the lab:

- One of our motors was broken at first. The noise in the gearbox indicated that it needed to be replaced, and the speed would not change despite changing the period and duty cycle for the motor driver, so we replaced it.
- After replacing the new motor, we found out that one of the wheels was not assembled straight compared to the others, which results a slight left turn when the tank moving. The speed for the wheels was also out of sync as one of the wheels would not vary much in speed, to the PWM was adjusted to match them as closely as possible.
- Because one of the wheels appears to be weaker than the other, there were issues implementing the turning. Usually, a wide turn would mean reducing the speed of the wheel on the side you want to turn. However, as indicated above we could not control the speed of the wheel very well.
- In addition, we kept the speed the same and tried to completely stop one wheel instead, to pivot the tank. Again, it would work on the stronger wheel side but not the weaker wheel, so we need to figure out how to adjust the wheel of the tank.

Because one side of the wheels are not moving properly, for the final demo we decided to forego the turning function and just let the tank move back and forth. Because the wheel speeds are not perfectly matched up, the tank does turn minorly on its own when moving forwards/backwards.

Distance Sensors/ADC pins

Difficulties we met during lab:

- We discovered after plugging in multiple sensors that the Beaglebone would fail to start up when SYS 5V was connected. Initially, we thought that this meant we had to change wires. This worked for a bit and then failed again. This error is probably a result of the system trying to draw too much power from the Beaglebone on startup.

We discovered that the system would boot up properly if we unplugged the SYS 5V from the breadboard, and plug it in after the Beaglebone had powered on. It would then continue to reboot properly after the 5V was reconnected. We decided plug/unplug for the demo.

Other Components

We initially attempted to get the bluetooth working for this lab. However, our bluetooth never entered the command mode. We debugged and finally found out that the timer set for it is only about 3 seconds so we cannot enter the command mode in time. We had to change to a new bluetooth but did not set it up in time for the demo as a result.

Conclusion

In this lab, we practised using the H-bridge motor driver to run motors and thus driving the tank. We also used timer interrupts to change the direction of the motor, by gathering data from distance sensors and interrupting the main program when the data was high enough to indicate an obstacle. This lab is the preparation of our final project and we have well prepared through this lab. Again, our debugging skills were developed through a series of hardware problems. One improvement possible is to add a remote controller via the bluetooth feature, and we will continue to work on fixing the right wheel/motor of the tank for the final lab so we can get the tank to turn properly.