MAZE SOLVING ROBOT

Fall 2018 Mid Term Report

Julie Kim (011963677)

CECS 490B Tuesday, Thursday 6:30pm – 8:20pm

Due Date: November 4, 2018

Contents

Maze Solving Robot—Micro Mouse

Team Member:

This project is contributed by one team member. All the work including research, simulation for software, hardware built, complete schematic and writing report are handle by one team member.

Biography:

Julie Kim is a senior student at California State Long Beach, majoring in Computer Engineering. Julie, she loves to learn and always try hard to independently tackle problem in real life. She like working on math problem because it has always been very easy to get good grade in class. She hates technology very much that make she wants to choose Engineer as her career path. She loves to clean, do house work and want to be able to do more heavy work just like most men do. She has developed a desire to exercise with heavy weight lifting to be fit and strong. When graduate and enter the engineering field, she wants to be mentally and physically ready to be a productive engineer team player.

Project Overview:

Maze Solving Robot project implement a robot that travel through a maze with standard size of 16x16 cell. Each cell is 17 Cm. The autonomous robot or called micro mouse travel through all the possible path it can find in the Maze and find its way to the goal, located at the center of the Maze, with the shortest path possible. Micro mouse is an autonomous robot that travel with the navigation of its sensor directing it to left turn, right turn, go straight or turn around when it encounters a dead end. The Micro Mouse make two runs. The first run, Micro Mouse learn all the path ways it finds when it reaches the goal. It recalculates the path and choose the shortest path possible to get to the goal of the Maze for its second run. Below shows the Micro Mouse travelling in a Maze.

Customer Needs:

Micro Mouse is a research product. As an engineering product, Micro Mouse is expected to be efficient in the robot three functionalities to travel in the center of the Maze, find the goal with shortest path, reduce power consumption to the minimum. Since the Maze is only 3x8 cells, the power used is enough for 2 Lithium 18650 of 3.7v and the travel time is between 5 to 10 minutes.

Project Specifications:

1. **Stopping collision and moving in center of the Maze:** three Ultrasonic sensors are used to navigate the robot through the Maze. One Ultrasonic sensor on the right detect the right wall, one on the left detects the left wall, and one at the front of the robot detects the front of the wall. When the front distance is less than 4cm, the robot stops. Since the sensor has a minimum distance of 2cm, 4cm can be accurately measured and prevent the robot from hitting the wall. The cell dimension is 17cmx17cm. The robot is 10cmx10cm giving the minimum space between it and the wall is 3cm for it to stay at the center of the Maze. 3cm is accurately detected by the sensor. The robot stirs right when its left wall distance is less than 3cm, and it stirs left when its right wall distance is less than 3cm. The confirmation to force the robot to stay in the center of the Maze is the different between left wall and right wall space is less than 1cm.
2. **Finding the goal with the shortest path:** to get the shortest path, the robot makes two runs. The first run uses the DFS algorithm to travel the entire Maze. DFS is implemented by going from on cell to the next. When the intersection is encountered, DFS prioritizes forward(F) as first option, right(R) as second option, and left(L) as third option. The algorithm stores all the chosen options and deletes all the incorrection options at the end of the program, so only the correction options of the turning directions (F, R, or L) are store in the array. As a result, in the second run, at all the intersections encountered, Micro Mouse gets the right direction from the array. It reduces the path that leads to dead end.
3. **Reducing power consumption to the minimum:** there are only 24 cells to run, so two lithium battery with total voltage of 7.4v reducing the need to have extra battery weight mounted on the robot. Less weight reduces power needed to run the robot from its stop position. Software code saves the power consumption by increasing speed gradually from 50% to 60% to 70% instead of speed changing from 10% to 90% which cost more battery power.
4. **Hardware—project components:** 10cmx10cm chaises robot car with two 5v DC motor attached, three Ultrasonic sensors receives and two H-bridges receiving 5v input each. One 32-bit MCU output 5v and source 500mA. Power supply input of 7.4v(2 lithium batteries) is regulated to 5v with max current of 1A to drive the MCU. All the input and output pins of Ultrasonic sensors and H-bridges can be driven by 8mA current from the MCU, so there is no transistor needed.

Work Breakdown Structure:

Power System	3.7v 2-Lithium Battery	LM2940CT-5v Regulator	Battery Holder	Power Software
DC Motor movement	2 Regular 5v Motors	L9110 H-bridge	Software Interrupt	
Distance Sensor	HC-SR04	Digital Input and Output	Software read and output digital	

32-bit MCU

Tm4c123ghpm

Development tool
Keil uVision5

JTAG Debugger onboard
MCU pins PC0-3

Software main loop constantly read digital distance sensor value

Front HC-SR04 read in digital distance value

Right HC-SR04 read in digital distance value

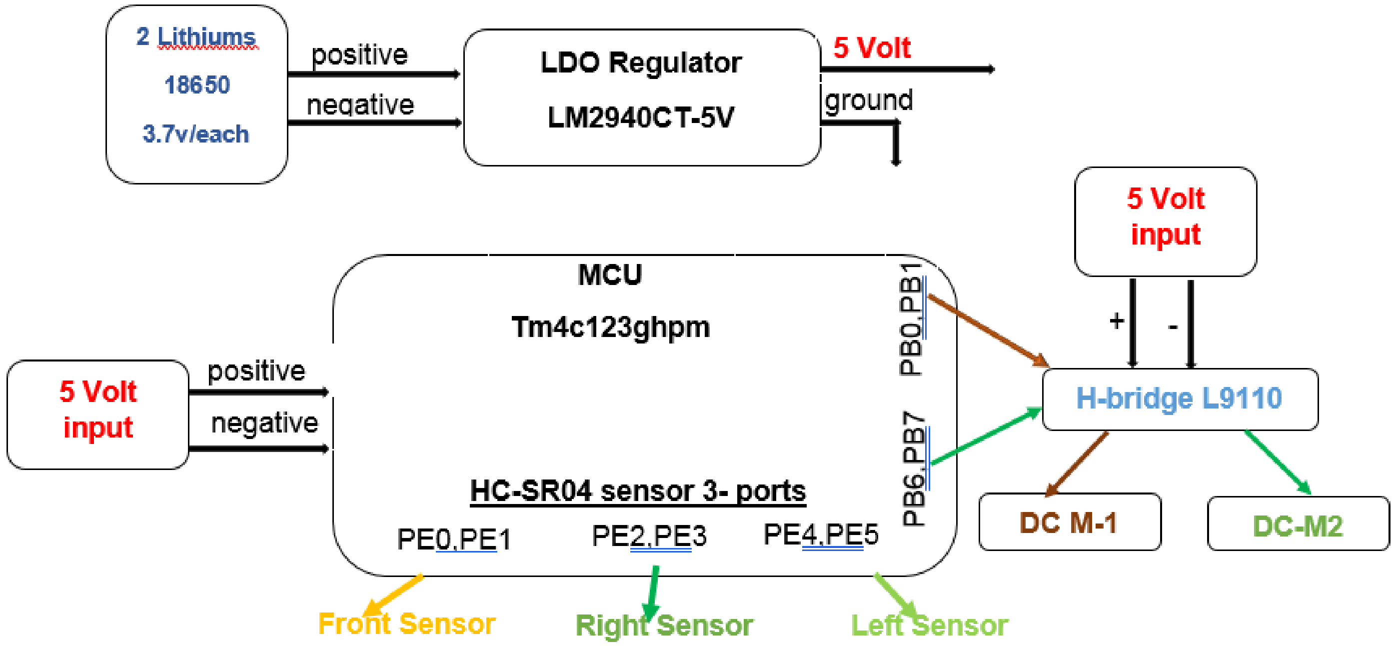
Left HC-SR04 read in digital distance value

Firmware implementing DFS algorithm using the digital distance value

Project Implementation:

The Micro Mouse autonomously travel in a 3x8 cells and 17cmx17cm each cell Maze. It starts entering the Maze, and it finds its autonomously find its way through the Maze to get to Maze goal. Ultrasonic sensor directs the movement of the Mouse to keep it track in the center of the Maze, acknowledge if there is wall on the left or right, and the sensor stops the robot from colliding into a wall. When the Mouse arrives at a cell that has two directions to go, it makes decision to turn left, right, forwards, backward or turn around, based on the predetermined choice forwards, right, left. Software application implementing DFS algorithm to search through all the path of the Maze (24 cells total). As it goes deeper and deeper to the Maze, if the Mouse get to the dead end, the Mouse turn around and back track to the previous path. As the Mouse travel cell by cell, the encountered intersections turning direction (R, L, F) is stored in the array by DFS algorithm. When the Mouse reaches the Goal, the array stores all the correct turning direction of the intersections of the Maze. For the second run, the robot uses the directions (F, L, R) stored in the array when it encounters the intersection to decide which way to go right(R), left(L), forward(F). According to the algorithm, the options are prioritized as F, then R then L. When the Mouse reaches the goal, it stops, and the implementation ends.

Overall System Functional Block Diagram:



Tm4c123ghpm				
Name	**Input/Output**	**Description**	**Voltage Level**	**Current**
VBus	Power Input	Supply Power to entir tm4c123 board	5.0 V	500mA
PA-PF	Both input or output, with digital and alternat function	Receiving input or produce output	3.3V	8 mA
PA2-PA7	Digital input	Receive input from HC-SR04	3.3v	8 mA
PB0, PB1, PB6, PB7	Digital output	Output to H-Bridge to control the Motor	3.3v	8 mA

Power Supply				
Name	**Input/Output**	**Description**	**Voltage Level**	**Current**
LM2940CT-5V	Regulator	Synchronous step down regulator with integrated fet	6-26v	1A
Sony 3.7v 2600mAh battery	Power supply	Output voltage to Regulator	3.7v	130 mA

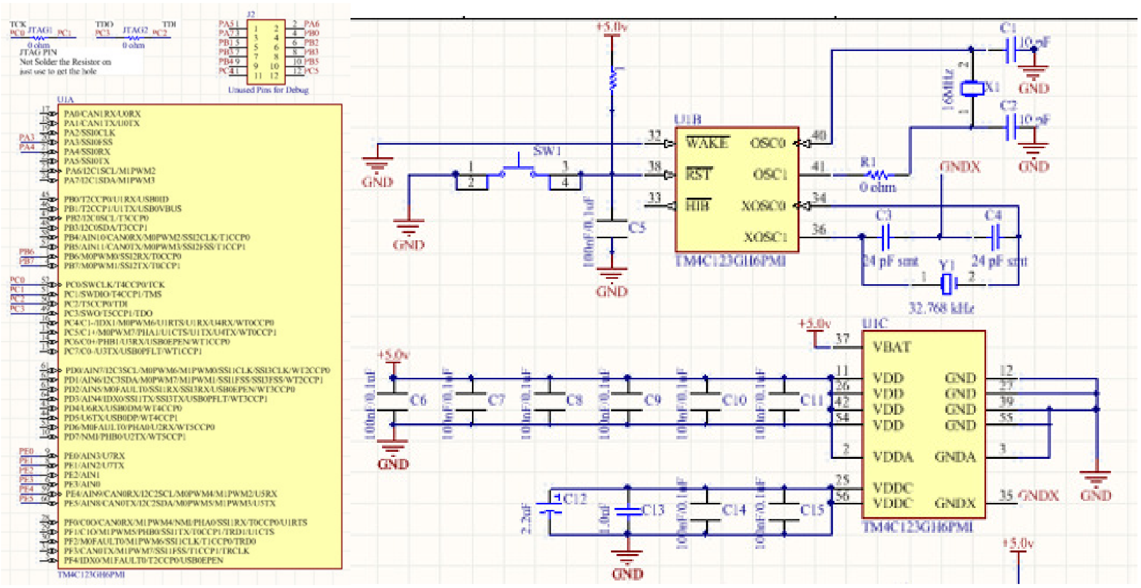
2 DC_Motors				
Name	**Input/Output**	**Description**	**Voltage Level**	**Current**
DC motor	Power Input	Run the motor wheels	5v	500 mA

HC-SR04 (Distance Sensor)				
Name	**Input/Output (digital)**	**Description**	**Voltage Level**	**Current**
Distance Sensor	Output Pulse	Send pulse time to microcontroller	5v	15 mA
Trigger pin	Input pin	Receive 10 us input from microcontroller	5v	15mA
Echo pin	Output pin	Output pulse between up to 25 ms	5v	15mA

H_Bridge(L9110)				
Name	**Input/Output(digital)**	**Description**	**Voltage Level**	**Current**
H-Bridge	Digital Input	Receive digital input from pins of the microcontroller	12v	2A
Speed(A-IA, B-IA)	Digital Input	Receive input from PB0 and PB1	5v	8 mA
Direction input (A-IB, B-IB)	Digital Input	Receive input from PB6 and PB7	5v	8 mA

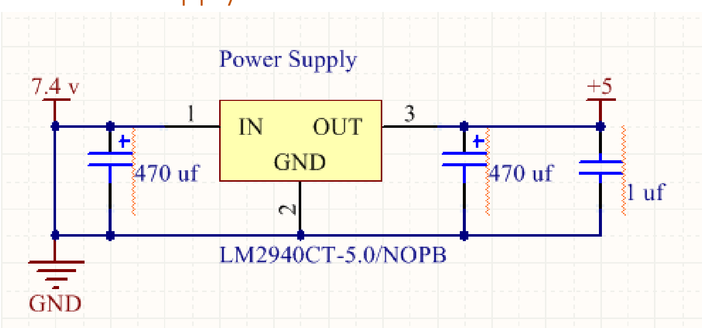
Subcomponent Descriptions:

❖ Tm4c123ghpm (MC):



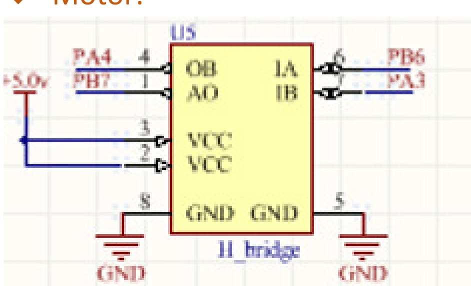
The microcontroller is 32-bit Arm processor with input power supply 5v, digital input and output pins. The debugging interface is JTAG. **JTAG:** tm4c123ghpm processor interface ready provided that is connected through port PC0(TCK), PC1(TMS), PC2(TDO) and PC3(TDI). It is used to debug the software.

❖ Power Supply:



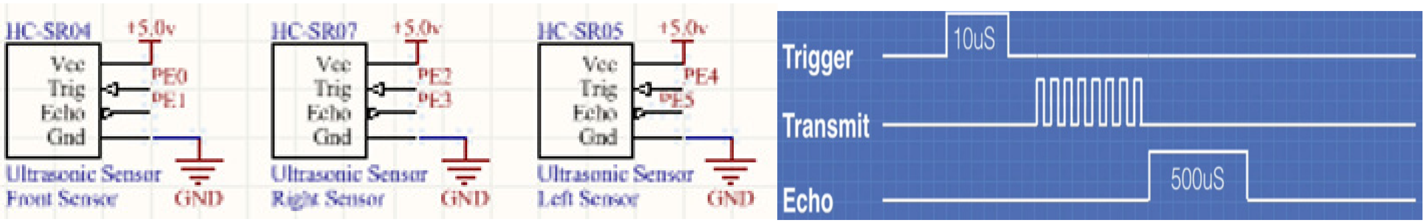
The LM0940CT-5v is positive regulator that can source 1 A of output current with a dropout voltage of 0.5v and maximum of 1 v.

❖ Motor:



Two motors receive DC voltage from the power supply to move the robot.

❖ Distance Sensor (HC-SR04):

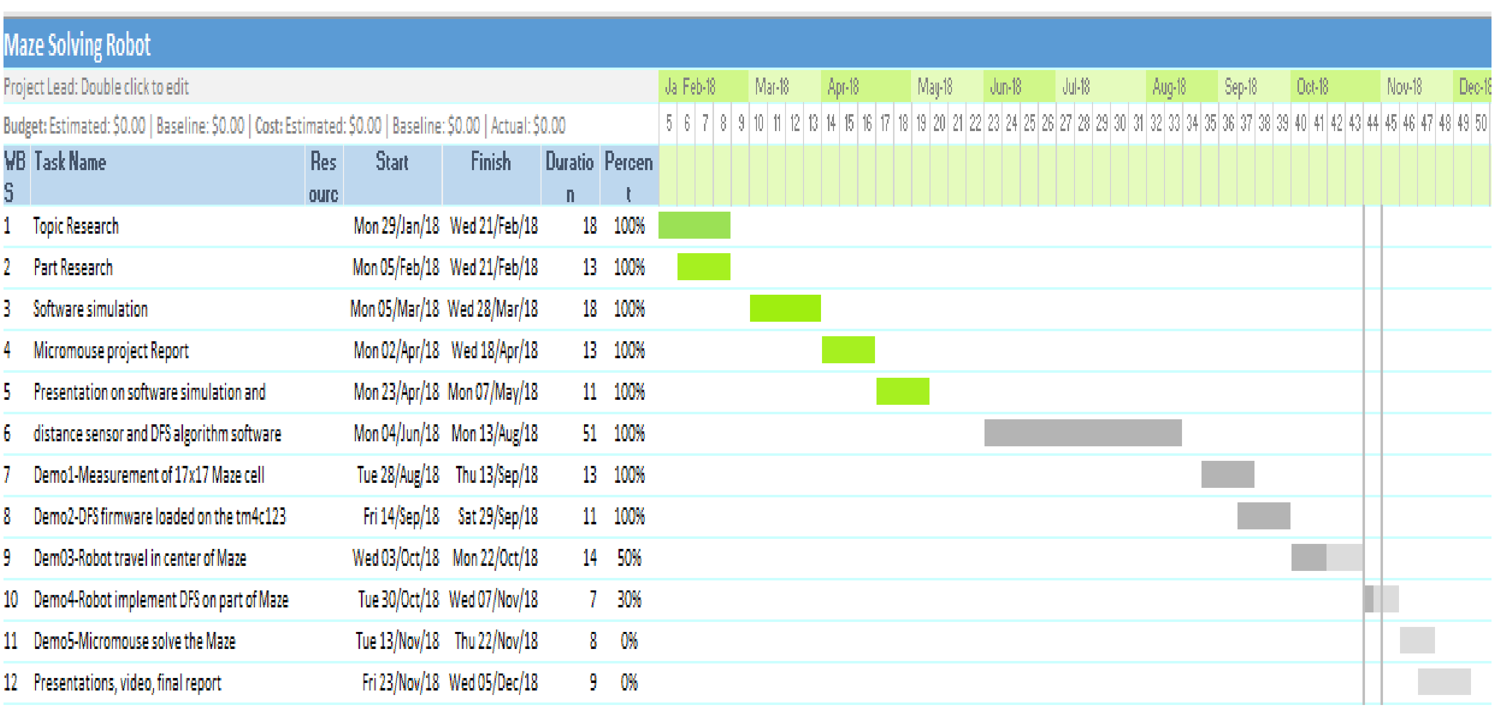


It capable of measuring distance between 2 cm to 400 cm. Offshell module with four pins, ground, vcc, trigger pin (input pin) and echo pin (output pin). If there is no object in front of the HC-SR04, the echo pin sends out a pulse of 38 ms. With object in front of it, the pulse time decrease. The pulse time and speed of sound is used to calculate the distance between the module and the object. The formula is: D=(pulse time/2)*speed of sound. In dry air of 200C, the speed of sound is 343 m/s. To communicate with the HC-SR04, the microcontroller sends out 10us pulse to trigger the module. HC-SR04 transmit 8 cycle burst of ultrasound at 40KHz and raise its echo. The echo pin output a pulse of 150 us to 25 ms depending on the proximity of the object to the module.

❖ H-bridge(L9110):

Control the direction and speed of the motor. The module has six pins. Two for speed, two for direction, VCC and ground. The first four pins are input pins from microcontroller. PB6 and PB7 of the microcontroller output the speed to pin A-IA, B-IA. PB0 and PB1 output the direction either 1 or 0 to pins A-IB and B-IB. The four screws on the module connected to the motor to spin the wheel.

Gantt Chart:



WBS	Task Name	Resourc	Start	Finish	Duration	Percent
1	Topic Research		Mon 29/Jan/18	Wed 21/Feb/18	18	100%
2	Part Research		Mon 05/Feb/18	Wed 21/Feb/18	13	100%
3	Software simulation		Mon 05/Mar/18	Wed 28/Mar/18	18	100%
4	Micromouse project Report		Mon 02/Apr/18	Wed 18/Apr/18	13	100%
5	Presentation on software simulation and		Mon 23/Apr/18	Mon 07/May/18	11	100%
6	distance sensor and DFS algorithm software		Mon 04/Jun/18	Mon 13/Aug/18	51	100%
7	Demo1-Measurement of 17x17 Maze cell		Tue 28/Aug/18	Thu 13/Sep/18	13	100%
8	Demo2-DFS firmware loaded on the tm4c123		Fri 14/Sep/18	Sat 29/Sep/18	11	100%
9	Dem03-Robot travel in center of Maze		Wed 03/Oct/18	Mon 22/Oct/18	14	50%
10	Demo4-Robot implement DFS on part of Maze		Tue 30/Oct/18	Wed 07/Nov/18	7	30%
11	Demo5-Micromouse solve the Maze		Tue 13/Nov/18	Thu 22/Nov/18	8	0%
12	Presentations, video, final report		Fri 23/Nov/18	Wed 05/Dec/18	9	0%

Functional Demos and Milestones Contract:

A thorough research was made in previous semester—CECS490A, and decided on the project topic, parts to use and initial start on the project. The project is Maze solving robot. The software of DFS algorithm was implemented and simulated in C++. The result was presented in previous semester.

The time between the semester was used to work on Ultra sonic distance sensor and DFS algorithm to load into the MCU-tm4c123.

Second semester—CECS490B, there are five demos needed to be made by working on the project one part a time.

Demo 1: Measurement of 17x17 Maze cell. Distance sensor to measure between the robot and the Maze wall. The measurement is in millimeter. Measurable Outcome: the distance is output on the UART in micrometer. The accuracy is +5 mm or -5 mm. The minimum measurement is 20 mm, the maximum is 55 mm and worst case is 65 mm. The minimum for front and right sensors is 20 mm. The minimum for the left sensor is 30 mm.

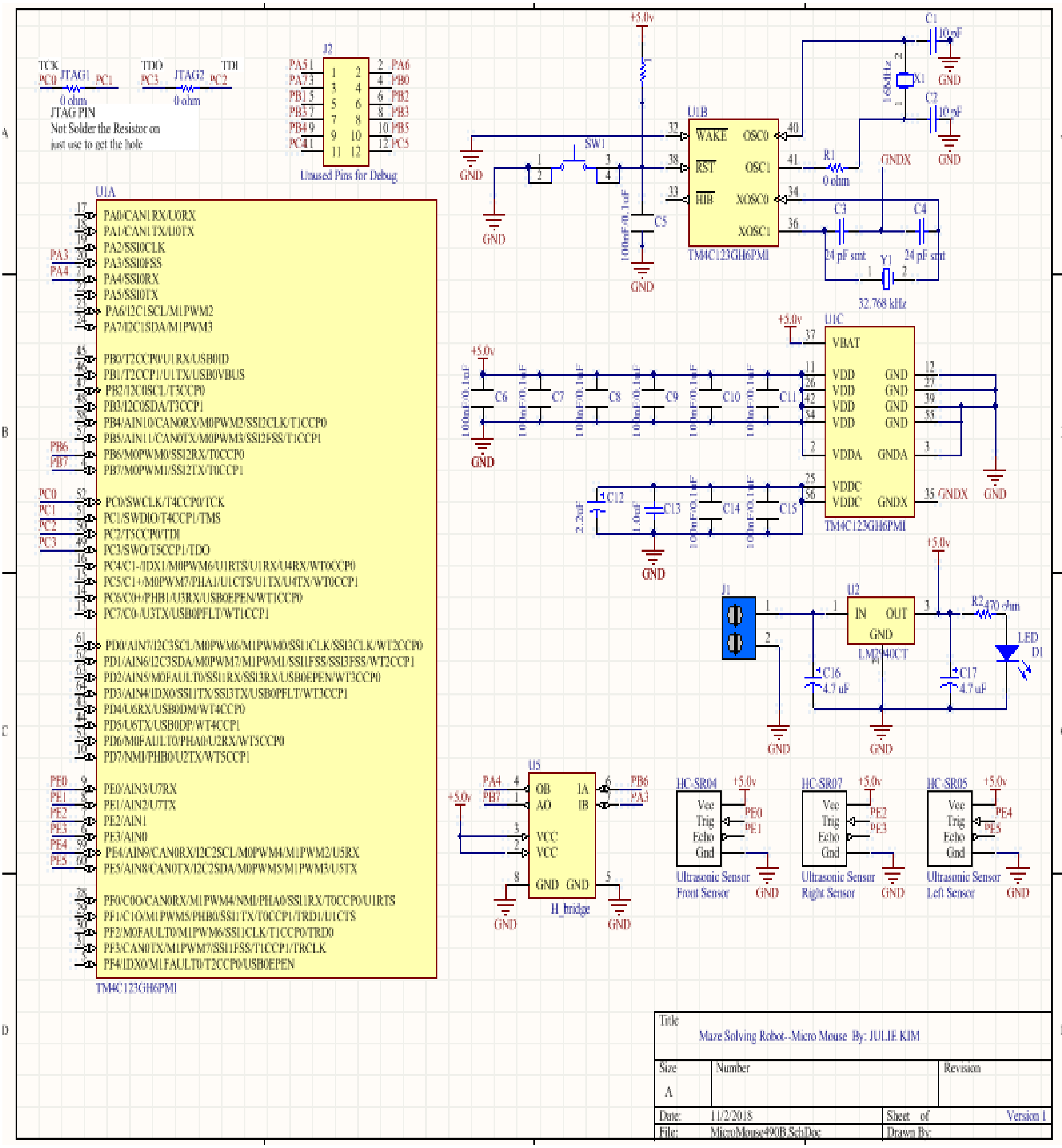
Demo 2: DFS firm ware loaded on the tm4c123 board.
Simulation of the Micro Mouse path and output the second path at the end of the program. UART is the terminal to be used.
Micro Mouse remember the correct turn on each intersection it encountered during first run. Outcome: the simulation firmware list the correct turn from the first run and intent to use this turn in the second run. The path is not listed, just the correct turn of each intersection used to get to the goal. The intersection that is dead end was deleted from the list.

Demo 3: robot follow the Maze wall, make U-turn, left turn and right turn, and it stay in the center of the Maze. Outcome: Micro Mouse travel in the center of the Maze.

Demo4: Actual run on the Maze. First run Micro Mouse implement the DFS algorithm to decide which turn to choose (Left, Right or forward) and where to go. When it reaches the goal, it stop and remember the correct turn of the previous intersection it encounter, and wait for the interrupt to start the second run. Outcome: Micro Mouse solve the Maze.

Demo5: Final demo of the Micro Mouse to solve the Maze. Outcome: Micro Mouse travel the Maze and find the shortest route to go.

Complete Schematic:



Bill of Material (BOM):

Description	Quantity	CompanyName	DesignItemId	Manufacturer Part No	Octopart Url	OctopartID
Res Metal Film 10K Ohm 1% 3/5W ±50ppm/°C Conformal AXL Thru-Hole Ammo Pack	1		CMP-0a5cc3a37cecd59b-6	LR1F10K		
Capacitor, Ceramic, 10 pF, +/-5 %, 200 V, 2-Pin THD, RoHS, Bulk	2		CMP-125554-4	C315C100J2G5TA	http://octopart.com/c315c100j2g5ta-kemet-92876	0f4cc64740e6d08b
CAP CER 24PF 50V 5% NP0 0603	2		06035A240JAT2A		https://octopart.com/06035a240jat2a-avx+interconnect+%2F+elco-3920530	16dddfece129c91a
C320 C 100nF Ceramic Multilayer Capacitor, 50 VDC, +85degC, Z5U Dielec, +/-20%	9		C320C104M5U5TA	C320C104M5U5TA	https://octopart.com/c320c104m5u5ta-kemet-39421137	9a2151723b9645e6
2200uF 35V	1		CMP-7a9bb39249a7a55c-2	UVZ1V222MHD		7a9bb39249a7a55c
MKS2 Series 63 V 1 uF ±10 % Radial Metallized Polyester Film Capacitor	1		MKS2C041001F00KSSD			
Cap, Alu Elec, 4.7uf, 100v, Rad	1		CMP-f90a3153675126cd-1	ESK475M100AC3AA	http://octopart.com/esk475m100ac3aa-kemet-22264637	f90a3153675126cd
Cap, Alu Elec, 4.7uf, 100v, Rad	1		ESK475M100AC3AA	ESK475M100AC3AA	http://octopart.com/esk475m100ac3aa-kemet-22264637	f90a3153675126cd
Red T-1 3/4 5 mm 30° 50 mcd 2.2 V Tinted Solid State LED Lamp Through Hole	1		CMP-f9af30a89dfb2668-1	WP7113ID	http://octopart.com/wp7113id-kingbright-868658	f9af30a89dfb2668
Euro Style 2 Position 5 mm 22-14 AWG Wire Receptacle Terminal Block	1		1776244-2	1776244-2	https://octopart.com/1776244-2-te+connectivity-40064487	ac786711dcf973bf
Board-To-Board Connector, Right Angle, WR-PHD Series, Through Hole, Header, 12, 2.54 mm	1		61301221021		http://octopart.com/61301221021-w%C3%BCrth+elektronik-32855539	
MBB0207 Series Axial Thin Film Resistor0Ohms +/-1% 0.6W +/-50ppm/degC	3		MBB02070Z0000ZCT00	MBB02070Z0000ZCT00		
Res Carbon Film 470 Ohm 5% 1/4W -400ppm/°C to 0ppm/°C Conformal	1		CF14JT470R			

Friday 18-May-18/2018 9:29:17 PM

Similar Products:

Micromouse competition is very popular in both US and Japan. Also in China and other part of the countries has it as competion and educational project. The lay out of the Maze routs are all different from place to place. The end point of Micromouse search can be the center of the maze or finding an exit of the Maze. The Maze construction can be two wall confined as path way or just a black line for the Micromouse to travel on.

The Maze construction for the competion is faily large and complicated, about 16 foot x 16 foot. This Microouse project follow the wall construction Maze. The Micromouse has to travel on the center of the path always and the end point is the center of the Maze. Since this Micromouse project is for educational purpose, the Maze construction is 3x8 cells and each cell is 17cmx17cm. The Maze is not large and not too complicated as the Maze mentioned. It might take the Micromouse only less than 10 minutes to travel the entire Maze.

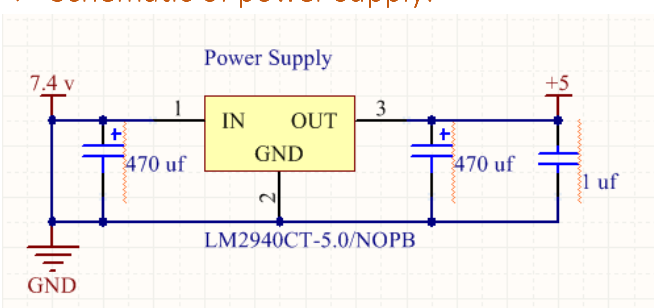
Although this project is very simple compared to the realy Micromouse competion, it is usefull for educational purpose. It teaches student how to design hardware to have enough powere to control the movement of the robot, how to interface the outside world to the embedded system, and how to solve problem in software and implement the solution in hardware. It is beneficial for teaching and learning. The example feature of Maze is below the actual one used in this project would be smaller and less path:

Societal and Environmental Impact or Importance:

Maze solving robot is built for education and competition purpose. As an engineer, students learn to cooperate knowledge and real-world product. Micro mouse autonomous robot is the combinational product of software and hardware implementation. Microcontroller contained the CPU interacts with the outside signal through the input and output pins to receive and send out information to navigate the Micro Mouse to be an autonomous. The software implementation of DFS is loaded into the MCU the information the Micro Mouse needed to make through the entire Maze and make decision of which direction to go.
The building of an autonomous product to solve the 3x8-cells with 17cmx17cm each cell Maze is an excellent exercise for engineer students to practice on the engineering work in solving real world problem by using computer.
The Micro Mouse light weight and small in dimension for low battery consumption.

Power Management:

❖ Schematic of power supply:

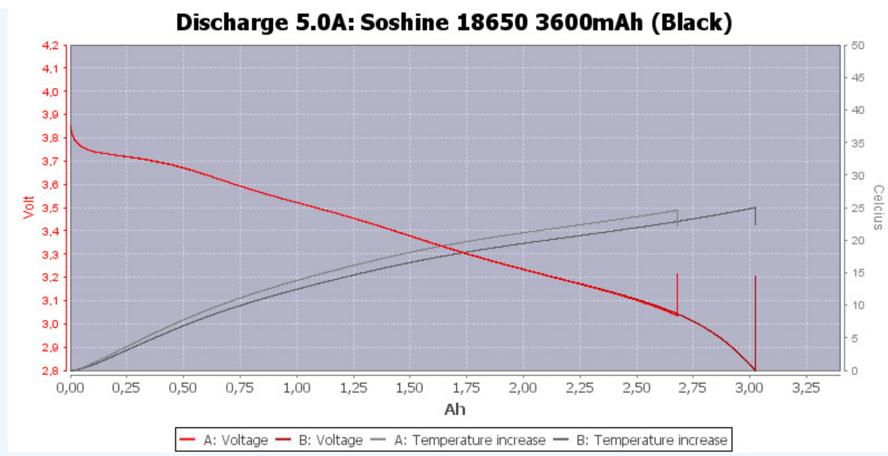


❖ Power Budget:

Component Name	#of Component	Current per Component	Current per Product	Supply Voltage	Power Supply	Power Total
Microcontroller tm4c123	1	500 mA	500 mA	3.3 v	1.65 w	
Regulator LM2940CT-5V	1	1A	1A	5v	5 w	
DC-Motor	2	2A	2A	5v	10w	
Ultrasonic Sensor	6	30 mA	180mA	5v	0.9w	
H-Bridge	1	1A	1A	5v	5w	22.55W

❖ Battery Discharge Curve:

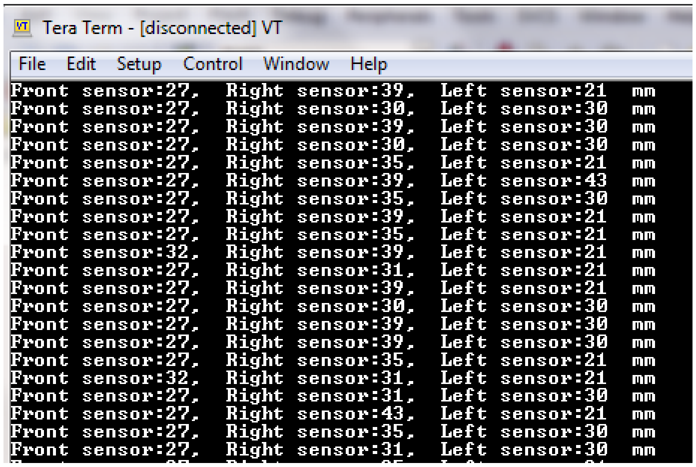
Two Lithium batteries (18650 3600mAh 3.7v) are used



Simulations and Software Verifications:

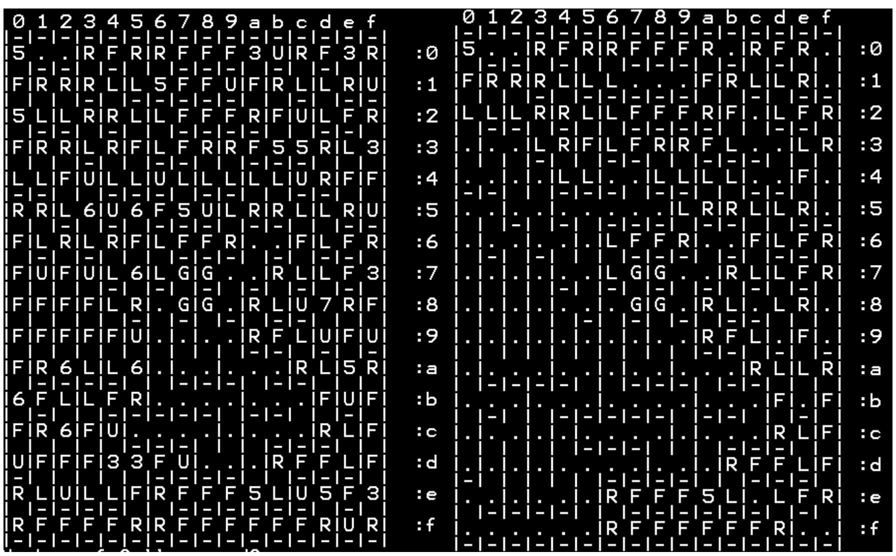
❖ Ultrasonic Distance Sensor hardware simulation:

PE0 sends out trigger pulse of 10 us to the sensor, PE1 receive input from the sensor pulse in millisecond to measure the distance between the module and the cell walls, the front-wall, right-wall and left-wall. It is more accurate as the object is father away. There some error when the distance is close to the module less than 2.5cm.



- **Depth First Search (DFS) Micro Mouse First and Second Run simulation (C++):**

The C++ software simulation of first and second run of the 16x16 maze. The goal of the Maze is at the center of the Maze and the simulated run start at the top left. In implementing the DFS algorithm, the first choice of the direction is forward(F), second direction is right(R), the last one is left(L). The "U" symbol indicates the route is a dead end. Different route is to be chosen. The Left image is the result of the first run, and the second is the second run, which take less cell to reach the goal.



- **Depth First Search (DFS) Micro Mouse Second Run on board simulation with UART:**

For the simplicity of verifying DFS algorithm is working on when loading on tm4c123, 24 cells only Maze is used to run the simulation. The follow array is hard coded as a route for the robot to follow.

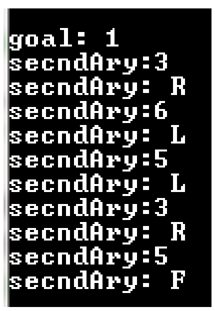
Maze [34] = {F, 0x03, R, R, U, 0x03, F, 0x06, 0x06, F, U, 0x06, R, R, U, 0x06, 0x06, R, R, R, 0x05, R, R, U, 0x05, F, R, 0x03, R, U, 0x03, 0x05, R, G}

The DFS code is loaded into the board, the robot moves according not on the simulated Maze but the direction (forward(F), right(R), left(L), u-turn(U)) of each array element. It looks like predetermined route for the robot to run, but it is not. Think about the way the robot moves is not matter of F, R, or L, the wall will determine its ways. The F, R, U, L are a random way to craft a imaginable simulated Maze. The important element of the array that need to be taken into consideration in implementing the DFS algorithm is the intersection that the robot encounter. According to the above 16x16 Maze, the cell that indicates—0x03, 0x05, 0x06—are the intersection. When the robot encounters these intersections, it needs to decide to turn right(R), forward(F), or left(L), u-turn(U) is the dead end which robot need to turn around. The intersections are the important data that determines how the Maze look like and how the robot itself will choose its direction to go by using the DFS algorithm to navigate itself. Therefore, the above array where the elements are F, R, or L is not import, but if the—0x03, 0x05, 0x06—intersections values are change that means **different Maze and different result of second run will be generated.**

According to DFS algorithm, the first choice of direction is forward(F), second choice is right(R), third choice is left(L). The array has more elements than the 24-cell Maze because when robot comes to dead end it needs to make a U-turn and back track the same route.

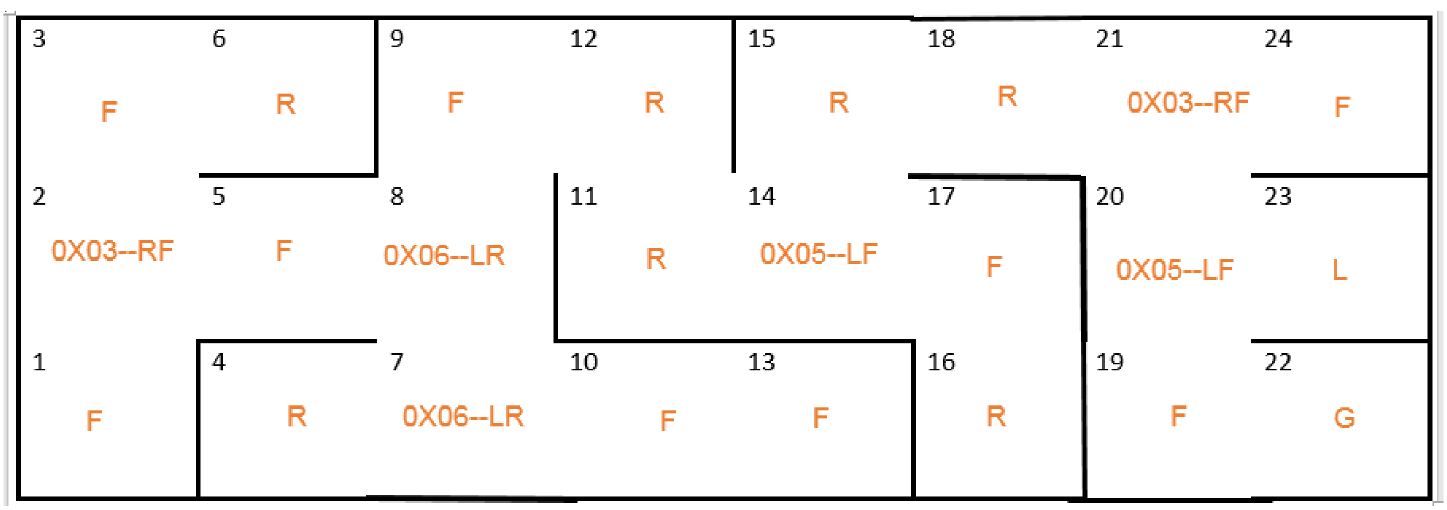
After the first running, the simulation presents the exact route (shorter route than the first run) for the second run.

The expected result to see is: **{0x03—R, 0x06—L, 0x05—L, 0x03—R, 0x05—F}**



the chosen option of direction when intersections are encounter

The diagram of the Maze according to the above hard coded array in the software program:



The intersections options for second run when the robot encounter them are determined, so based on the Maze diagram at cell-2 is R-turn, cell-8 is L-turn, cell-14 is L-turn, cell-21 is L-turn, cell-20 is F-turn.

Complete Software Code:

```
//standard C library with <>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <limits.h>

#include "UART.h"
#include "PLL.h"
#include "PWM.h"
#include "SysTick.h"
#include "Timer0.h"
#include "Timer1.h"
#include "Timer2.h"
#include "tm4c123gh6pm.h"

unsigned long distance0=0, distance1=0, distance2=0;
unsigned long count0=0, count1=0, count2=0;

long StartCritical (void);    // previous I bit, disable interrupts
void EndCritical(long sr);    // restore I bit to previous value

void WaitForInterrupt(void);
void DisableInterrupts(void);
void EnableInterrupts(void);
void PortF_Init(void);
void PortE_Init(void);
void PortA_Init(void);
void PortB_Init(void);
void PortC_Init(void);
void PortD_Init(void);

void timer0FrontDistance0(unsigned long *distance0);
void timer1RightDistance1_1(unsigned long *distance1);
void timer2LeftDistance2_2(unsigned long *distance2);
void stirrRight(void);
```

```
void stirrRight(void);
void stirrLeft(void);
void stop(void);

void UserTask(void){     //Timer0
  if (GPIO_PORTE_DATA_R&0x10)
    count0 = count0 + 1;
}

void UserTask1(void){    //Timer1
  if (GPIO_PORTD_DATA_R&0x04)
    count1 = count1 + 1;
}

void UserTask2(void){    //Timer2
  if (GPIO_PORTE_DATA_R&0x04)
    count2 = count2 + 1;
}
void OutCRLF(void){
  UART_OutChar(CR);
  UART_OutChar(LF);
}
```

```
int main (void) {
  unsigned long index = 0, i = 0, ii = 0;
  unsigned char tt = 0, t_index = 0,stp = 0;
  unsigned char r=0, l=0, f=0;
  unsigned char DFS_1stRun;
  unsigned char turn = 0, deadEnd=0, goal=0;
  unsigned char arrayTurns[4] = {0, 0, 0, 0};
  unsigned char travalArray[250];
  unsigned char secndRunArray[250];

  unsigned char tempArray[33] = {'F',0x03,'R','R','U',0x03,'F',0x06,
                                  0x06,'F','U',0x06,'R','R','U',0x06,
                                  0x06,'R','R','R',0x05,'R','R','U',
                                  0x05,'F','R',0x03,'R','U',0x03,0x05,
                                  'R'};
  PLL_Init();                    // bus clock at 80 MHz
  PortF_Init();
  PortE_Init();
  PortA_Init();
  PortB_Init();
  PortC_Init();
  PortD_Init();
  SysTick_Init();
  UART_Init();
  Timer0_Init(&UserTask,80);    // 1us interrupt
  Timer1_Init(&UserTask1,80);   // 1us interrupt
  Timer2_Init(&UserTask2,80);   // 1us interrupt
  PWM0A_Init(40000,15000);      // initialize PWM0
  PWM0B_Init(40000,15000);      // initialize PWM0

  GPIO_PORTF_DATA_R = 0x04;     // Blue led
  GPIO_PORTB_DATA_R |= 0x01;    // MotorA foward
  GPIO_PORTB_DATA_R |= 0x02;    // MotorB foward
  PWM0A_Duty(1000);
  PWM0B_Duty(1000);

  stop();
  GPIO_PORTF_DATA_R = 0x04;     // Blue led

  for (unsigned char t=0; t<250; t++)
    travalArray[t] = 0;
  for (unsigned char s=0; s<250; s++)
    secndRunArray[s] = 0;

  UART_OutString("InString---> Distance is: ");
  OutCRLF();

  while (goal == 0) {
    while (turn != 'U') {
      while (turn == 0) {

        timer0FrontDistance0(&distance0);
        timer1RightDistance1_1(&distance1);
        timer2LeftDistance2_2(&distance2);

////Test Distance Sensor
        while(1) {
          GPIO_PORTF_DATA_R = 0x02;      //RED led, Timer0
          GPIO_PORTB_DATA_R |= 0x01;     //MotorA, foward
          GPIO_PORTB_DATA_R |= 0x02;     //MotorB, forward
          PWM0A_Duty(35000);             //10%   MotorA, foward
          PWM0B_Duty(35000);             //10%   MotorB, foward
          //Timer0 Port PE4-5, distance0
          timer0FrontDistance0(&distance0);
          //Timer1 Port PD2-3, distance1, 1
          timer1RightDistance1_1(&distance1);
          //Timer2 Port PE2-3, distance2, 2
          timer2LeftDistance2_2(&distance2);
```

```
          UART_OutString("Front sensor:");
          UART_OutUDec(distance0);
          UART_OutString(",  Right sensor:");
          UART_OutUDec(distance1);
          UART_OutString(",  Left sensor:");
          UART_OutUDec(distance2);
          UART_OutString("  mm");
          OutCRLF();
          SysTick_Wait10ms(10);
        }

        GPIO_PORTF_DATA_R = 0x00;
        if (t_index == 33)
          turn = 'G';
        else {
          turn = tempArray[t_index++];
          if ((turn > 0x0F)&&(turn != 'U')) {
            turn = 0;
          }
        }
        if (turn == 'G') {
          goal = 1;
          turn = 'U';
        }
        else if (turn == 'U') {
          deadEnd = 1;
          turn= 0;
        }
        UART_OutString("****turn: ");
        UART_OutUDec(turn);
        OutCRLF();
      } ////END of while(turn == 0)

      if ((deadEnd == 1)&&(goal == 0)){
        turn = 'U';
        goal = 0;
      }
```

```
      else if ((deadEnd == 0)&&(goal == 0)){
        if (turn == 0x03){////***turn RF == 0x03
          travalArray[index++] = 0x03;
          travalArray[index++] = 'R';
          travalArray[index++] = 'F';
          turn = 'F';
          turn = 0;
          f = 1;
        }
        else if (turn == 0x05) {/////***turn LF ==0x05
          travalArray[index++] = 0x05;
          travalArray[index++] = 'L';
          travalArray[index++] = 'F';
          turn = 'F';
          turn = 0;
          f = 1;
        }
        else if (turn == 0x06) {/////***turn LR == 0x06
          travalArray[index++] = 0x06;
          travalArray[index++] = 'L';
          travalArray[index++] = 'R';
          turn = 'R';
          r = 1;
          turn = 0;
        }
        else if (turn == 0x07) {////***LRF == 0x07
          travalArray[index++] = 0x07;
          travalArray[index++] = 'L';
          travalArray[index++] = 'R';
          travalArray[index++] = 'F';
          turn = 'F';
          f = 1;
          turn = 0;
        }
      }
    }
    /////END OF:"while(turn!='U')"; now turn == 'U', goal==0;
```

```
////deadEnd==1; goal==0; turn=='U'; go to TOP STACK, get the 2ND TURN
if (goal == 0) {
  while (turn == 'U'){
    deadEnd=0;
    DFS_1stRun = 'U';
    for (i=0; i<4; i++)
      arrayTurns[i] = 0;
    i = 0;

    ////array that hold the turns:0x03,0x05,0x06,0x07
    if ((travalArray[index]) == 0)
      index--;

    /*////pop off the TOP stack and save values to array[4];
    ////0x07==LRF-->[0]=='F', [1]=='R', [2]=='L',[3]==0x07
    ////0x06==LR--->[0]=='R', [1]=='L', [2]== 0, [3]==0x06
    ////0x05==LF--->[0]=='F', [1]=='L', [2]== 0, [3]==0x05
    ////0x03=RF---->[0]=='F', [1]=='R', [2]== 0, [3]==0x03*/
    while (((DFS_1stRun>0x0F)&&(index>0))||
           ((DFS_1stRun>0x0F)&&(index==0))) {
      DFS_1stRun = travalArray[index];
      travalArray[index--] = 0;
      if (DFS_1stRun > 0x0F){
        arrayTurns[i] = DFS_1stRun;
        i++;
      }
    }
    arrayTurns[3] = DFS_1stRun;

    ////RF,LF=0x03,0x05..........................................
    if ((arrayTurns[3] == 0x03)||(arrayTurns[3]==0x05)) {
      if ((arrayTurns[0]=='F')&&(arrayTurns[1]=='R')) {
        deadEnd = 0;
        turn = 'L';
        index++;
        travalArray[index++] = arrayTurns[3]; //push 0x03
        travalArray[index++] = arrayTurns[1]; //push 'R'
      }
      else if ((arrayTurns[0]=='F')&&(arrayTurns[1]=='L')){
        deadEnd = 0;
        turn = 'R';
        index++;
        travalArray[index++] = arrayTurns[3]; //push 0x05
        travalArray[index++] = arrayTurns[1]; //push 'L'
      }

      /*////2ND TURN, deadEnd = 1;
      ////0x05==LF--->[0]=='L',[1]==0,[2]== 0,[3]==0x05
      ////0x03=RF---->[0]=='R',[1]==0,[2]== 0,[3]==0x03 */
      else if ((arrayTurns[0]=='R')||(arrayTurns[0]=='L')){
        deadEnd = 1;
        if (arrayTurns[0]=='R')
          turn = 'L';
        else if (arrayTurns[0]=='L')
          turn = 'R';
      }
    }
    ////END OF: 0x03==RF, 0x05==LF.................................

    ////RL=0x06....................................................
    else if (arrayTurns[3]==0x06) {
      ////printing travalArray************************************
      if (arrayTurns[0]=='R') {
        deadEnd = 0;
        turn = 'F';
        index++;
        travalArray[index++] = arrayTurns[3]; //push 0x06
        travalArray[index++] = arrayTurns[1]; //push 'L'
      }
      else if (arrayTurns[0]=='L') {
        deadEnd = 1;
        turn = 'R';
      }
    }
    ////END OF: 0x06==LR...........................................
```

```
        if (turn == 'R'){
          r = 1;
          turn = 0;
        }
        else if (turn == 'L'){
          l = 1;
          turn = 0;
        }
        else if (turn == 'F'){
          f = 1;
          turn = 0;
        }
      }////END OF:"while(turn=='U')"; now turn == 0, turn != 'U'
    }////END OF if(goal==0)
  }////END OF  while(goal == 0), Reach the GOAL,  goal == 1

  UART_OutString("goal: ");
  UART_OutUDec(goal);
  OutCRLF();

  i = 0; ii = 0;
  while (travalArray[i] != 0){
    if ((i>0)&&(travalArray[i-1]>0x0F)&&
       (travalArray[i]>0x0F)) {
      secndRunArray[ii-1] = travalArray[i];
    }
    else {
      secndRunArray[ii] = travalArray[i];
      ii++;
    }
    i++;
  }

  ii = 0;
  while(secndRunArray[ii] != 0) {
    UART_OutString("secndAry:");
    if (secndRunArray[ii] == 70)
      UART_OutString(" F");
    else if (secndRunArray[ii] == 82)
      UART_OutString(" R");
    else if (secndRunArray[ii] == 76)
      UART_OutString(" L");
    else
      UART_OutUDec(secndRunArray[ii]);
    OutCRLF();
    ii++;
  }
  ii = (ii+1)/2;
  OutCRLF();
  UART_OutString("number of Turns secndArray: ");
  UART_OutUDec(i);
  UART_OutString("---");
  UART_OutUDec(ii);
  OutCRLF();
  if (goal == 1){
    stop();
  }
}
//////END OF Main**********
```

```
/*
goal: 1
secndAry:3
secndAry: R
secndAry:6
secndAry: L
secndAry:5
secndAry: L
secndAry:3
secndAry: R
secndAry:5
secndAry: F
*/
void stop(void) {
void stirrLeft(void) {
void stirrRight(void) {
//Timer0 Port PE4-5, distance0
void timer0FrontDistance0(unsigned long *distance0) {
//Timer1 Port PD2-3, distance1, distance11, 1, 11
void timer1RightDistance1_1(unsigned long *distance1) {
//Timer2 Port PE2-3, distance2, distance22, 2, 22
void timer2LeftDistance2_2(unsigned long *distance2) {
void PortF_Init(void){
//PB6, PB7 is PWM, PB0, PB1 is Direction (PB6-->PB0, PB7-->PB1)
void PortB_Init(void) {
void PortC_Init(void){
void PortD_Init(void){
void PortE_Init(void){
void PortA_Init(void){
```

Reference:

Timer Example: http://users.ece.utexas.edu/~valvano/arm/Timer0A.c

http://users.ece.utexas.edu/~valvano/index.html

http://users.ece.utexas.edu/~valvano/arm/