Buggie log

This is the first attempt to organize and combine all components for the buggie.

Here is a list of files containing source code:

1. us.c/h
2. uart.c/h
3. motor.c/h
4. SysTick0.c/h
5. Main.c

**us.c/h**

These two files contain the code to manipulate the ultrasonic sensor. The initialization functions need to be called before using the measuring functions. The time stamp will start recording time after the initialization function, and, for current implementation, after interruption is enabled. The measure function will return distance in cm. The time stamp function will return time in minute and second.

void Clock\_init();

This function initializes the clock signal

void TIMERA0\_init();

This function initializes the TIMERA0 for recording time\_stamp

void US\_init();

This function initializes pins for trigger ultrasonic sensor and record echo signals. It uses pin 4.6 as trigger and pin 4.7 for echo. It also registers IRQ for receiving interrupt from rising or falling edge at pin 4.7

long measure\_dist();

This function measures the distance between the ultrasonic sensor and the object in front of it. It triggers the sensor for several times and return the average value in cm. The specific size of the sample can be modified by changing the macro in us.h. The range of distance is (0, 300)

char\* get\_time\_stamp();

This function returns the time passed since the start of the measurement (marked by US\_init). The time stamp is record as “min:sec” in string.

**uart.c/h**

These files contain the code to manipulate the UART port. UART0 is at the micro-USB port. UART2 is at pin 3.2 and pin 3.3. These two ports allow serial communication between the microcontroller and the console on the computer.

void UART0\_init();

This function initializes UART0.

int UART0\_puts(const char \*str);

This function sends string through UART0 by spin waiting.

void UART2\_init();

This function initializes UART2.

int UART2\_puts(const char \*str);

This function sends string through UART2 by spin waiting. The transmit pin is pin 3.3. It also adds a carriage return ‘\r’ to every line feed ‘\n’.

int UART2\_gets(char \*str);

This function receives string through UART2 by spin waiting. The receive pin is pin 3.2. It returns after the receiving line feed ‘\n’ or end of the string ‘\0’

**motor.c/h**

These files contain the code to control the motor through H-Bridge. It use pin 2.7/2.6 for the right wheel, pin 2.5/2.4 for the left wheel. It allows the buggie to move forward and backward. It turns by letting two wheels turn in different directions. Currently it only allows the wheels to turn at a fixed speed. SysTick0.c/h are required since they provide functions for delay.

void motors\_init();

This function initializes pins for motors

void forward(int time);

This function sets both wheels forward and pause after a time in ms.

void backward(int time);

This function sets both wheels backward and pause after a time in ms.

void turn(int angle); // Turn buggy by an angle in degrees

This function turns buggy in an angle from -180 to 180. The constant for degree per ms in rotation is defined as macro.

**SysTick.c/h**

These file contain functions for creating delay using SysTick

void SysTick\_Delayms();

This function delays in microseconds.

**main.c**

The main function contains the state machine of the buggie. It initializes all components like UART, motors, ultrasonic sensors. It waits for a ‘go’ signal from the Bluetooth serial input and starts moving. Currently, it moves around like a spiral and collect distance data at each turn.

#include "msp.h"

#include <stdio.h>

#include <string.h>

#include "uart.h"

#include "motor.h"

#include "us.h"

#define unit\_step 1000

enum buggieState {

    INIT,

    IDLE,

    MOVE,

    SCAN

};

void main(void)

{

    WDT\_A->CTL = WDT\_A\_CTL\_PW |             // Stop watchdog timer

            WDT\_A\_CTL\_HOLD;

    char readbuf[20];

    char writebuf[20];

    int step = 1;

    long distance;

    char\* time\_stamp;

    enum buggieState currState = INIT;

    while(1)

    {

        switch(currState)

        {

        case INIT:

            Clock\_init();

            motors\_init();

            UART2\_init();

            TIMERA0\_init();

            US\_init();

            UART2\_puts("INIT\n");

            currState = IDLE;

            break;

        case IDLE:

            UART2\_gets(readbuf);

            sprintf(writebuf, "%d\n", strncmp(readbuf, "go",2));

            UART2\_puts("buggie started");

            if(strncmp(readbuf, "go",2) == 0)

            {

                \_\_enable\_irq();             // Enables interrupts to the system

                currState = MOVE;

            }

            break;

        case MOVE:

            forward(unit\_step \* step);

            step++;

            currState = SCAN;

            break;

        case SCAN:

            distance = measure\_dist();

            time\_stamp = get\_time\_stamp();

            sprintf(writebuf, "%s, %d\n", time\_stamp, distance); // format the output string

            UART2\_puts(writebuf);

            turn(-120);

            currState = MOVE;

            break;

        }

    }

}