Sergio Guerrero
CSE 3442- Embedded Systems 1
Final Project Report
December 6, 2021

Final Project Report

Introduction:

In this project we were to both build and configure a moving robot using the various techniques used in class lectures. We used Sensors behind the motor which triggered a pulse to the sensor indicating that a single turn was made. This was a 46:1 motor which meant that every 46 spins of the smaller wheel, was equal to 1 spin of the output wheel being the larger one. Some techniques used include configuring GPIO pins, using a Uart to allow to print and the user to input, configuring Wide Timers to keep track of time intervals as well as being used as a counter, and PWM's which allowed us to configure the two motors driving the robot and set them to different values.

Theory Of Operation:

One of the first steps in this project was done in Lab 6 as we were to take the TV controller and find the NVIC code as well as the data bits code for every button. This will allow us to later program each button to whatever we desire.

Next was building and soldering the robot in a proper fashion where all the components mounted on top of the robot were connected to their respective pins and could still all be powered by the battery pack of 4 double A batteries.

Then followed the step to find out which button was which from the remote by using a timer check the separation between pulses. Each button had a unique code which could be identified depending on the time interval between pulses. A 1 bit had an interval of 2T and a 0 bit had a interval of 1T. This was we could identify each bit on each button, and this identify which button was pressed. For this part I originally used the Wide Timer 0, but later had to switch to Wide Timer 3 since Wide Timer 0 was being used to for the Sensors behind the motor.

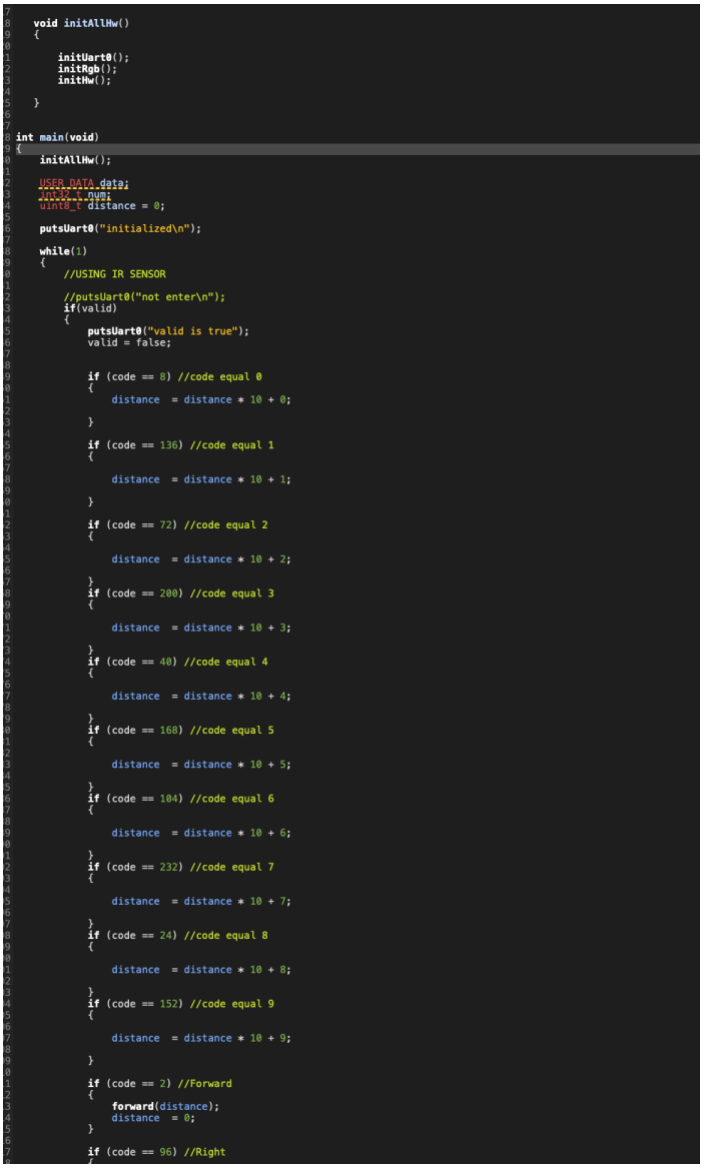
Once we were able to identify each button, then we moved onto configuring the PWM's for both motors. I used the PWM 1 module with both the 2 and 3 generators being configured to both motors. Main part of this step was to configure the PWM's properly to move the motors thus moving the wheels. As well as using Wide Timers for the sensors and program them to count up after each magnet passed over the sensor, which indicated the wheel had moved 1/46 of its circumference. For the sensor outputs, I configured them to PC4 and PC6 on the red board. By doing so I was able to configure the TAV in each Timer to count up as the magnet passed the sensor on each spin. This gave me control of exactly of far the motors should spin.

Last step of the process consisted of combining on the previous steps from before and configure them to work in sync with each other. By using the TV controller, I was able to use the IR sensor mounted to the robot to identify which button was being pressed. Calculate the distance the user wanted to move, and use a value in the sensors, to stop the motors once that value has been reached. Then finally call the appropriate function to turn on the motors using the

pwm and finally have the robot move. An example of this can be if the user pressed the “1” button then the “0” button on the remote, then pressed the “up” button, then the robot would move 10 cm forward. Another example would be if the use pressed “9” then “0” the. “right” button, then the robot would rotate to the right at an angle of 90 degrees. Next was to do these same steps but instead of using the remote, do it with the Uart command from the terminal. If the user enters “forward 10”, then the robot would move 10cm forward, and if the user entered “right 90”, then the robot would rotate to the right at an angle of 90 degrees.

Code:

Below is a small snippet of code from the final Lab 9. This is the code which calls the functions, forward(), right (), and left(). To move depending on which buttons were pushed. And the second image is the code (which is commented out but still runs) which calls the move functions depending on the commands entered on the Uart0 terminal. To use the TV remote, we must leave the Uart0 command code commented out so that there is interference. And vice versa, to use the Uart command, the IR sensor code in lab 9 should be commented out.



```
void initAllHw()
{

    initUart0();
    initRgb();
    initHw();

}


int main(void)
{
    initAllHw();

    USER_DATA data;
    uint8_t num;
    uint8_t distance = 0;

    putsUart0("initialized\n");

    while(1)
    {
        //USING IR SENSOR

        //putsUart0("not enter\n");
        if(valid)
        {
            putsUart0("valid is true");
            valid = false;


            if (code == 8) //code equal 0
            {
                distance  = distance * 10 + 0;

            }

            if (code == 136) //code equal 1
            {

                distance  = distance * 10 + 1;

            }

            if (code == 72) //code equal 2
            {

                distance  = distance * 10 + 2;

            }
            if (code == 200) //code equal 3
            {

                distance  = distance * 10 + 3;

            }
            if (code == 40) //code equal 4
            {

                distance  = distance * 10 + 4;

            }
            if (code == 168) //code equal 5
            {

                distance  = distance * 10 + 5;

            }
            if (code == 104) //code equal 6
            {

                distance  = distance * 10 + 6;

            }
            if (code == 232) //code equal 7
            {

                distance  = distance * 10 + 7;

            }
            if (code == 24) //code equal 8
            {

                distance  = distance * 10 + 8;

            }
            if (code == 152) //code equal 9
            {

                distance  = distance * 10 + 9;

            }

            if (code == 2) //Forward
            {
                forward(distance);
                distance  = 0;
            }

            if (code == 96) //Right
```

```
        if (code == 96) //Right
        {
            cw(distance);
            distance  = 0;
        }
        if (code == 224) //Left
        {
            ccw(distance);
            distance  = 0;
        }


    }


    //USING UART CMDS

   // if (kbhitUart0())
  //  {

    putsUart0("Enter command: ");
    getsUart0(&data);
    parseFields(&data);

    if (strcmp(&data.buffer[0], "forward") == 0)
    {
        num = getFieldInteger(&data, 1);
       forward(num);
    }
    else if(strcmp(&data.buffer[0], "right") == 0)
    {
        num = getFieldInteger(&data, 1);
        cw(num);
    }
    else if(strcmp(&data.buffer[0], "left") == 0)
    {
        num = getFieldInteger(&data, 1);
        ccw(num);
    }
  //}


}
```

Conclusion:

In conclusion, this project, although lengthy, was a good learning and hands on experience for me. Learning about configuring the GPIO and setting each bit individually was a good way to learn about the tedious and time consuming such project like this could be. One of the more interesting parts of this project was configuring the PWM to work and rotate the motors. To see the pulse width modulation in effect with the different values we input was good as we were able to control the speed of each motor individually. There were some hiccups along the way which added to the difficulty of the project. The majority of these were on the hardware side of the project as there were many occasions where there was an unintended short, thus burning out component and having to replace each one.