ButterBOT Base/RC Car

Project Number: 8

Dr.Schmalzel

Nathan Sulzer and Andrei Creanga

12/20/17

Introduction:

Over the course of this class we have become familiar with programing the outputs/inputs of our MSP430 processors, working with serial communication, and interacting with signals from the outside world with ADCs. Now we are at the point where we can put them together to create functioning open and closed loop systems that enable us to create functional electronic devices. Our project consists of the base for what will eventually become a butter passing robot based on a hit television show. Currently our system is open loop allowing UART serial communication to input commands to the car along with using ADCs that read from an analog joystick. Using both of these functions we can allow for control of the car by a user input and eventually an additional circuit that we intend to reuse from the original tank.

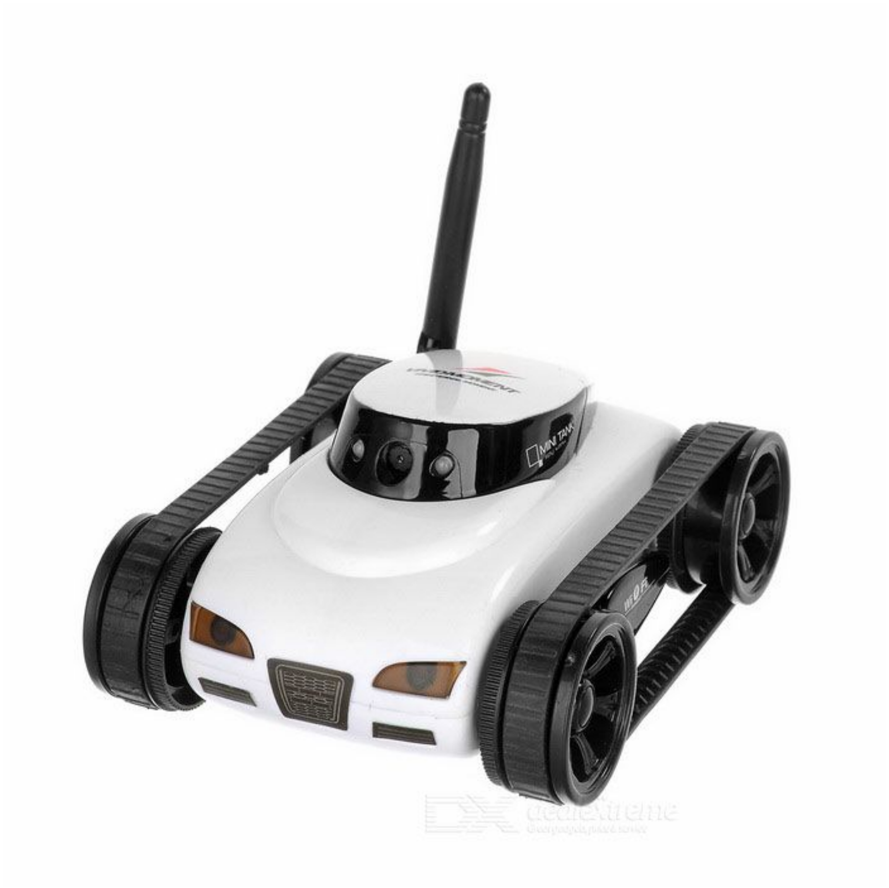


Figure 1: Ispy mini tank that was used as starting point for robot

The basis of our project was the Ispy Mini RC tank which has a few features we believed would be useful to our final project. The operation of the the RC tank was split into 2 seperate circuits one controlled the driving of the RC tank and the other controlled the camera and wifi communication of the RC tank with its phone app.

Design Specification:

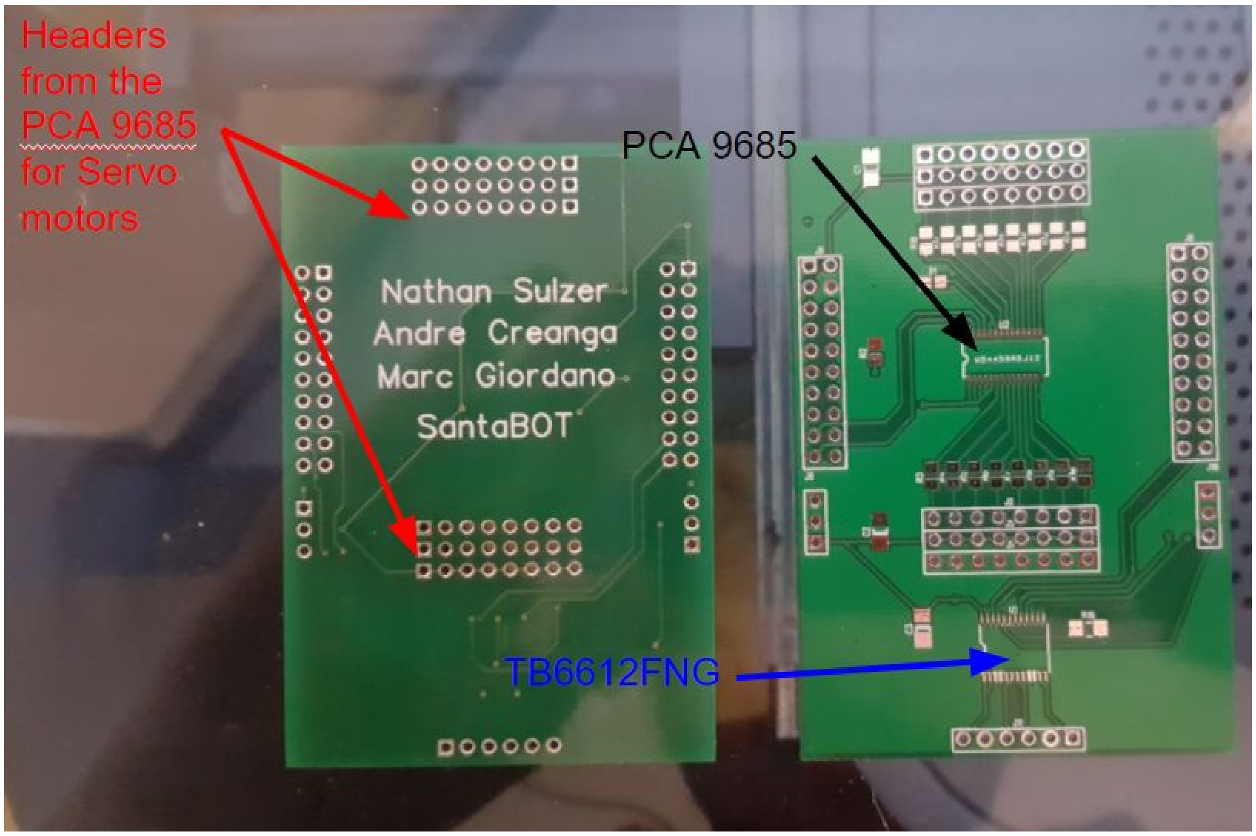


Figure 2: PCB layout

Our design originally included a PCB Daughter card in order to simply the connections of the Launchpad to the surface mount motor drivers however due to missing the Motor Driver ICs for the motor drivers in the final parts order. Talking specifics of the PCB, our PCB was simply a way to connect the desired pins of the MSP430FR6989 Launchpad to the enable pins of the PCA 9685 and the TB6612FNG. The PCA 9685 also needed to be attached to the I2C pins of the 6989 as the chip operates based on I2C to change the PWM value of up to 16 different output pins. The PCA 9685 is designed for the purpose of driving different colors for LEDs as were implemented in the milestone 1 assignment. The purpose of the PCA 9685 was to drive servo motors that would create the grabbing function for our ButterBot which only required additional pins on the board for the servo ground and power then we tied the LED PWM pin to the servo control. A 220 ohm resistor was required for protection of the servo motors from the output of the PCA 9685 which was recommended by another PCB that used this chip for the same purpose.

Since we couldn’t get the PCB finished in time for the final demo we switched over to a proof of concept for our base driving functions with the L293D motor driver and a miniature breadboard but the parts necessary to finish the PCB have been ordered so that our next model can make use of the PCB and hopefully enable us to have a working ButterBOT at some point in the future. The connections on the PCA 9685 consisted of connecting the address bits to ground because we had only 2 devices that needed to utilize I2C then each of the PWM outputs was tied to a header in the set of 3 that can be seen above or below our central chip which is the PCA 9685. The lower chip at the bottom of the board is the TB6612FNG which is what is directly replaced in our demo design by the L293D as it would have controlled 2 DC motors using an dual H bridge as the L293D would have the main difference between TB6612FNG and the L293D is that TB6612FNG has PWM inputs that allow for the motors to be turned either faster or slower in the same fashion our fan could be controlled in milestone 2.

The base of our robot contained four wheels, two of which had motors attached, that were wired up to the L293D motor driver and controlled by the MSP430FR6989 microprocessor. Attached to the MSP430FR6989 was a Fuel Tank MKII Battery BoosterPack Plug-In Module by TI, to power the car this module also has the capabilities to be used as an I2C device allowing us to display the battery level and other information on the 6989s LCD display but these features were not completed by the time of the demo but they are being worked on for future updates to the project. Also included was a RQS25 joystick used to control direction of the motor driver as well as voltage applied to it.

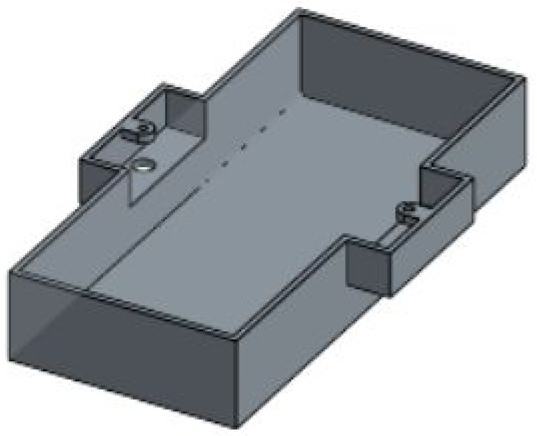


Figure 3: 3D printed shell

Figure 3 above shows the 3D printed model used to create a useable platform for the databoard and circuitry to rest on when being used with the ispy mini base. Space was allocated to contain the motor drivers as well as screw directly into the original screw holes alloted in the original ispy mini base. This part was printed and used to resolve the issue of the databoard not being set and constantly moving when the RC bot was in motion as a result of not being fixed to the base.

A majority of the code used was based on our Milestone 2 code due to familiarity with the pins used. In the case of a UART connection only one movement command can be implemented over UART at a time (Forward, Reverse, Rotate Left, Rotate Right). This actually gets implemented using hexadecimal which is the easiest way of sending numbers to the device over UART. Reverse is implemented using the internal design of the L293D driver which will invert the polarity of the high and low input signal to the motor based on the 2 input pins. The L293D was connected to each motor independently, allowing one wheel to rotate forward while the other rotates back. Rotate Left and Right is done by using the L293D to only reverse 1 motor direction at a time while the other motor would turn in the forward direction.

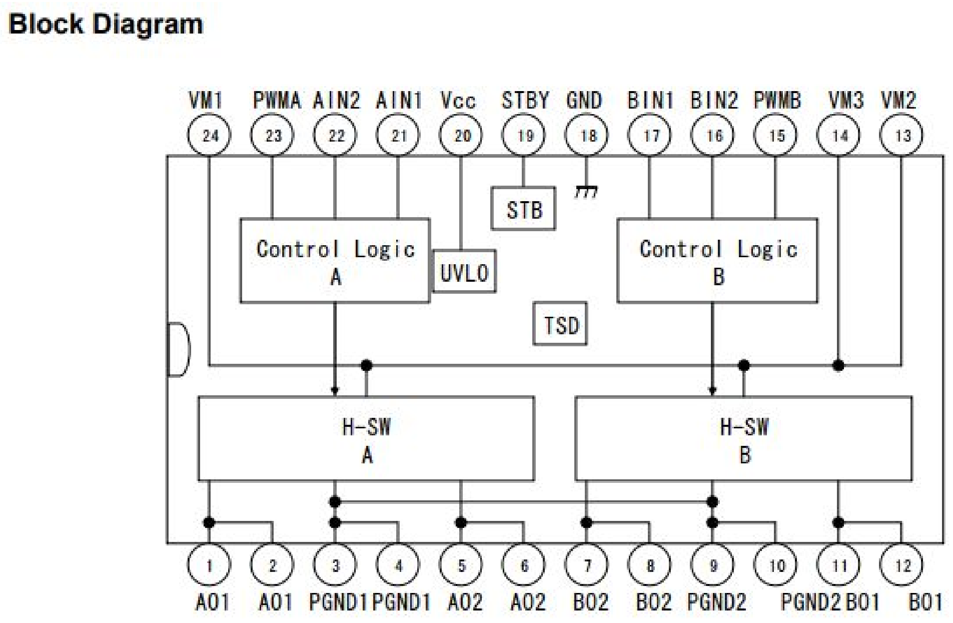


Figure 4: TB6612FNG PINOUT

For the original design of the bot that would be implemented by using the same basic implementation for our motor driver but the IC that was to be used was the TB6612FNG which featured the same basic inputs of the L293D motor driver but also featured inputs for PWM that would allow for the speeds of each motor to be changed to faster or slower and 2 inputs allowed for this change to happen independently on one another.

Figure 4 above shows the pinouts of the TB6612FNG which was originally supposed to be implemented into our robot base and the image below shows the IC that we used to replace it for the final demo.

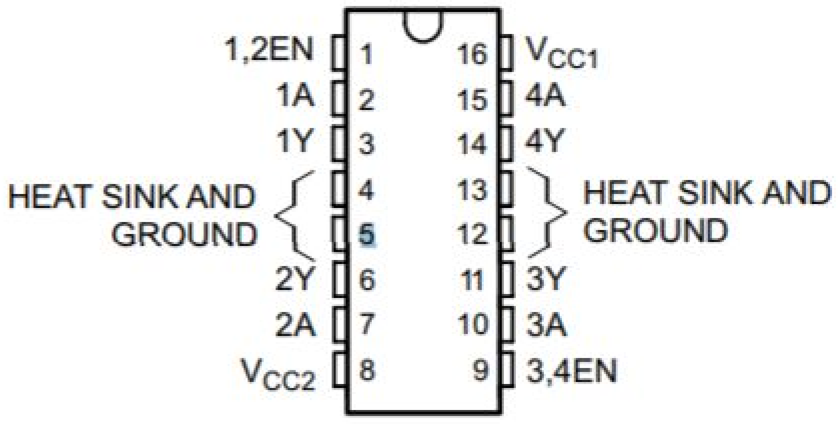


Figure 5: Pin layout for L293D Motor Driver

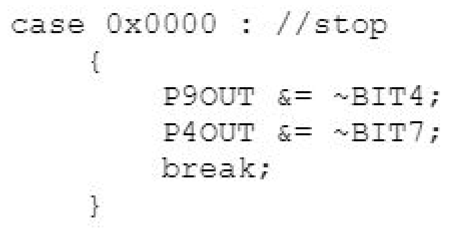
Figure 5 shows the pin layout of the motor driver uses to drive both motors as well as switch directions of operation of each motor. The main purpose of this choice of motor driver was that it used 2 input pins that would determine direction of the motor just as the TB6612FNG had.

Code Design:

The UART currently operates by using the following commands to change the direction of the car currently these are just placeholders to test out that our code can functionally change the cars direction using commands input from UART:
If all goes according to plan, then the car should move in the direction specified by the value input by the user over UART. The car will continue to move in that direction until a new direction is inputted, or it is told to stop.

The analog and digital commands being used to control the car had a conflict in control with our original design so we needed to add the last 2 functions in order to allow for us to switch which peripheral was in control at a given time this made it possible for our analog joystick to control it at times and our UART commands to control it when the UART was turned off.

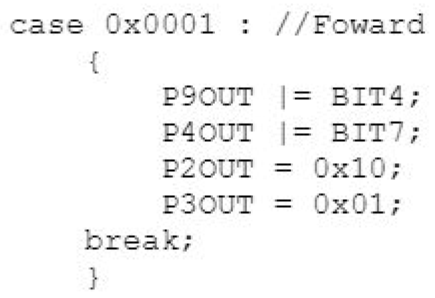
Our UART functionality was implemented through a case statement which allowed us to simply make the cases of the UART input change the motor driver inputs as discussed above. Pins 9.4 and 4.7 were connected to the enable pins of each side of the L293D driver so our stop function simply programed them to be off as shown in the code in figure 6.



```
case 0x0000 : //stop
     {
         P9OUT &= ~BIT4;
         P4OUT &= ~BIT7;
         break;
     }
```

Figure 6: Case statement for stopping

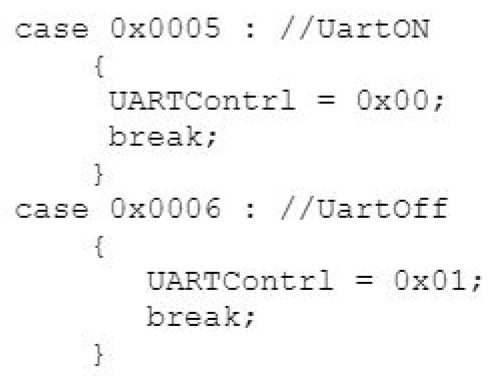
The code for programing a direction to the motors involved changing pins 2.3 and 2.4 for the direction of the right motor and 3.0 and 3.1 for the direction of the left motor. An example of our code doing this can be seen below in the case statement segment dedicated to the forward direction for our UART control.



```
case 0x0001 : //Foward
     {
         P9OUT |= BIT4;
         P4OUT |= BIT7;
         P2OUT = 0x10;
         P3OUT = 0x01;
     break;
     }
```

Figure 7: Case statement for forward

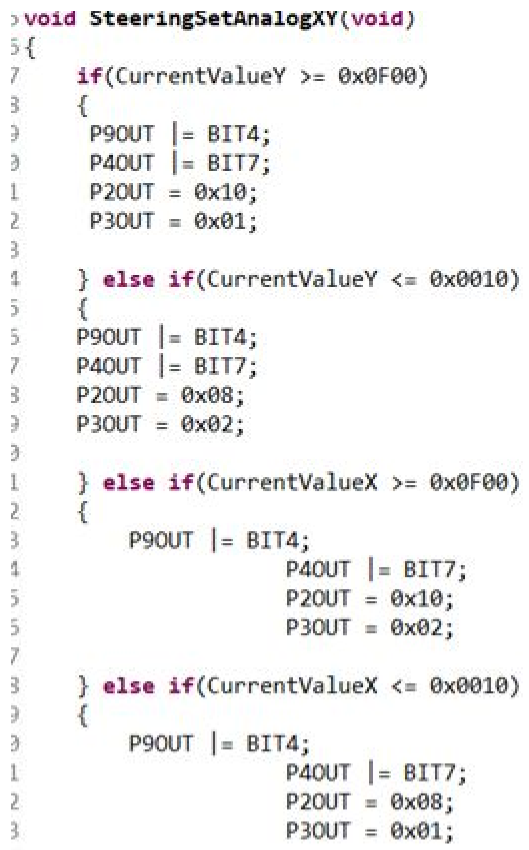
The code for the other 3 directions follows that same premise for operation as can be seen in the code itself. The only cases that vary from that basic idea are the cases controlling the UART on and off which set a variable called UARTContrl which would turn on/off an while statement that held the code for the analog joystick reading in it. The cases for the UART control can be seen below in Figure 8.



```
case 0x0005 : //UartON
     {
      UARTContrl = 0x00;
      break;
     }
case 0x0006 : //UartOff
     {
        UARTContrl = 0x01;
        break;
     }
```

Figure 8: Case statements for UART control

The joystick: The joystick is an analog system of control directly wired into the msp430fr6989 board. Being an analog system it requires ADCs in order to communicate with the digital logic of the 6989 which was done using 2 ADCs which read the x and y values of our joystick and convert them to the directions of the L293D motor controller. The code below shows the if statement that determines the output value of the analog joystick and outputs the proper direction to the L293D motor driver.



```
void SteeringSetAnalogXY(void)
{
    if(CurrentValueY >= 0x0F00)
    {
     P9OUT |= BIT4;
     P4OUT |= BIT7;
     P2OUT = 0x10;
     P3OUT = 0x01;

    } else if(CurrentValueY <= 0x0010)
    {
    P9OUT |= BIT4;
    P4OUT |= BIT7;
    P2OUT = 0x08;
    P3OUT = 0x02;

    } else if(CurrentValueX >= 0x0F00)
    {
        P9OUT |= BIT4;
                    P4OUT |= BIT7;
                    P2OUT = 0x10;
                    P3OUT = 0x02;

    } else if(CurrentValueX <= 0x0010)
    {
        P9OUT |= BIT4;
                    P4OUT |= BIT7;
                    P2OUT = 0x08;
                    P3OUT = 0x01;
```

Figure 9: Joystick analog controller code

Processor Selection

The choice to use the MSP430FR6989 was an easy decision for us to make as It was the most advanced of all our boards and had an additional value of being used in our milestone 2 code which meant we had a lot of prior coding experience to fall back on when it came to this board and many of our fellow classmates were using the same microprocessor in their final project which allowed for us to seek assistance from each other in getting the project completed and working which was a major help when it came to small coding issues or looking for examples of I2C implementation.

Future Improvements

One of the large improvements that I want to make to this project is to add in the original camera from the Ispy mini along with the wifi and camera driver that are on the PCB for the camera this PCB can be seen in figure 8. The Wifi chip is used to communicate with the custom Ispy mini APP and the other chip is used to decode the camera output and send it to the app so the camera feed can be viewed from a cell phone. Both these chips only have SPI capability as far as communication goes and that leads me to believe that the 4 output wires of the PCB are likely Sclk, MOSI, MISO and Selection to allow this board to communicate with the motor driver board of the original Ispy mini.

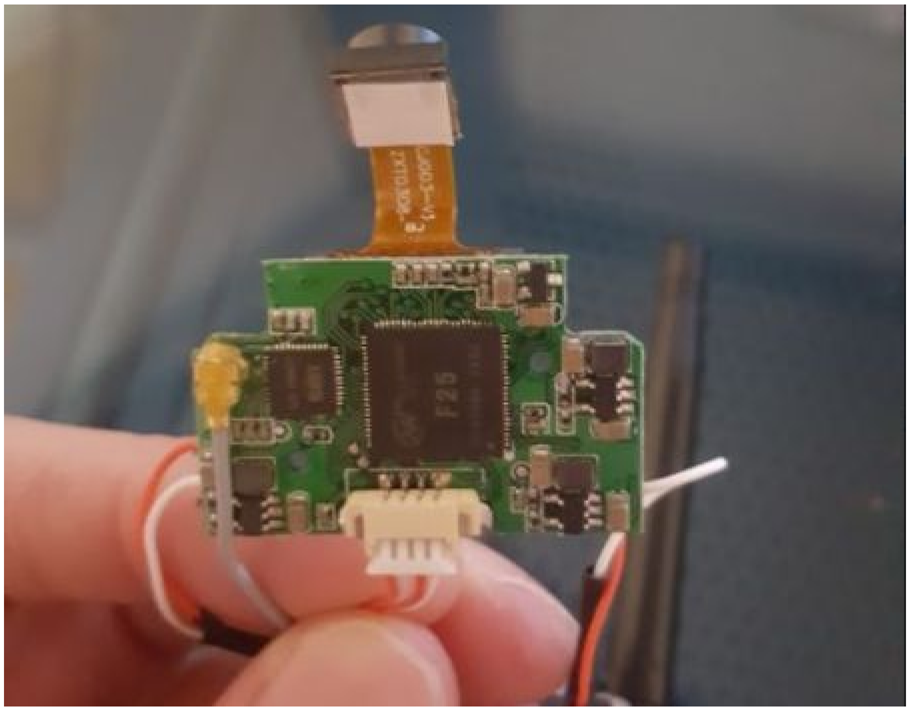


Figure 9: Driver board for the camera and wifi features of the Ispy mini

Over the break I intend to look deeper into these 4 output lines and check for whether they are in fact SPI communication lines and if they are I will reuse them to communicate with the SPI lines of the FR6989.

Results and Conclusion:

When presenting, our presentation went off almost perfectly. We had some last second issues regarding the battery not working for the presentation demo due to low battery level but after some charging everything worked well. The car was successfully able to perform all of the required actions including moving forward and backward, turning left and right, and stopping. This project has given us a look into the greater application of embedded systems and hopefully sometime soon this full robot will be finished and our PCB can be implemented into the final design that we complete.

References:

PCA9685 Datasheet
https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf

TB6612FNG
https://www.pololu.com/file/0J86/TB6612FNG.pdf

MSP430FR6989 Datasheet:
http://www.ti.com/lit/ds/symlink/msp430fr6989.pdf

L293D Motor Driver Datasheet:
http://www.ti.com/lit/ds/symlink/l293.pdf