Lab Report 4

Interrupts and Realtime

EE 474 Fall 2016
Dr. Allan Ecker
11/17/2016

Bryan Bednarski (1337864)
Justin Allmaras (1329197)
Colin Summers (1332139)

1.0 Introduction

The Lab 4 assignment was designed specifically for teams to implement their motor control H-bridge with pulse width modulation (PWM) motor control inputs to control variable motor speeds. This process allows the user to pass various duty cycles to both motors to controls degrees of motor output in either direction. This capability allows teams to move the rover forwards, backwards, left and right under their control. In addition to PWM motor controls, we also developed a general, timer interrupt system, to communicate distances interpreted by four IR sensors mounted to each side of the rover, with the motor control protocol in place. The timer interrupt is critical because it samples and reports analog values read from the IR sensors. These values are read as analog voltage outputs from the IR sensors, but converted to readable integer values by analog to digital converters (ADCs) built into the Beaglebone Black. Interrupts are critical to the development of all robotics projects, especially those that implement various transducers for real-time interaction with the environment. They typically operate at a high level of functional priority, and are termed the name “interrupt” because they can set flags within the general system to alert it of critical situations and needs for adjustments.

In addition to setting up the ADC pins for interrupts, and developing a corresponding PWM motor control system, we also built in a joystick control and over-voltage protection for the ADC pins. Implementing the joystick is a similar process to the sensors, but ADC values are much less noisy as the feedback to the ADC pins is simple a voltage determined by the current position of two-axis potentiometers. To protect the ADC inputs to the beaglebone, we built a op-amp clamp between the output of each sensor and input to the Beaglebone. This clamp limited the voltage range from 0 to 1.8 volts to prevent the Beaglebone ADC pins from burning out. With this hardware in place, we finally developed a functional control loop to control the motor controls with the joystick, preventing it from crashing into objects with STOP interrupts from the sensors.

2.0 Design Specification

This section details the design process taken to complete this lab. The following subsections describe the implementation of the motor driver, infrared sensors and joystick, respectively.

2.1 Motor Driver

In order to drive the tank, a motor driver circuit is needed. We used the TB6612FNG motor driver integrated circuit. See Appendix A for the application diagram of the motor driver. Interfacing with the Beaglebone required 5 GPIO outputs and 2 PWM outputs. The driver uses two GPIO pins determine the direction of each motor and uses 1 PWM signal to control the speed of the motors. The driver also uses one GPIO pin as the standby which must be set high in order to enable the outputs from the H-Bridge to the motor. The truth table for interfacing with the driver is shown in Figure 1.

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

Figure 1: TB6612FNG Motor Driver Truth Table

We used the default period of 500000 ns for the pwm signal and varied the duty cycle from 0% to 100% to change the speed of the motor. We also kept the default inverted polarity of the PWM driver Based on testing, we found that our motors began to stall out at around 50% duty cycle, so we are using that as the minimum duty cycle that we use to run our motors.

Our motor driver code is configured such that it can run with or without interrupt signals from the infrared sensors that are attached to our rover. When the infrared sensors are used, the robot will not allow movements that are likely to cause collisions, helping eliminate the possibility of human driver error causing damage to the rover.

2.2 Infrared Sensors

On our rover, we mounted 4 infrared proximity sensors to detect objects near the rover in front, back or on either side. We used Sharp 2Y0A21 sensors for this purpose. According to the datasheet, the sensor has a read range from 10 cm to 80 cm. In our experimenting, we found this to be accurate for our sensors. Figure 2 shows the voltage/distance curve for the sensors.

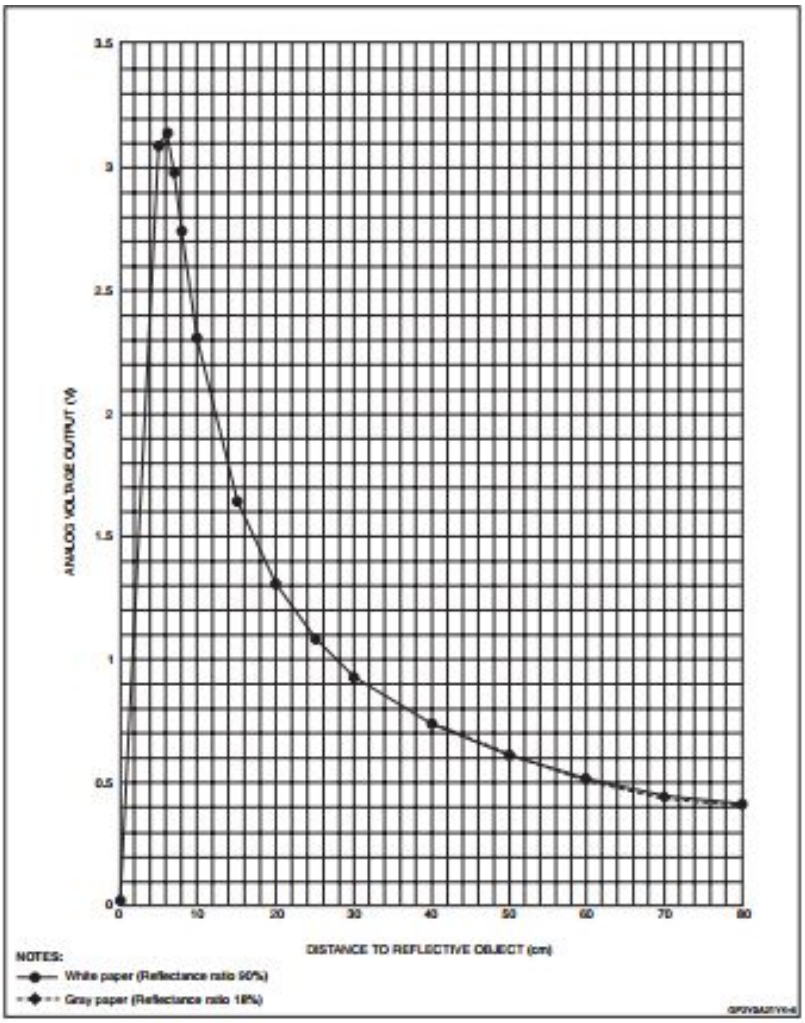


Figure 2: Output Characteristics of IR Sensor

From the AM335 datasheet, the maximum input value for the analog input pins depends on whether or not the internal voltage reference or the external voltage reference is used. From the Beaglebone schematic, see Figure 3, the external reference is used, so the the full-scale input range is from VREFN to VREFP, which is 0V to 1.8V.

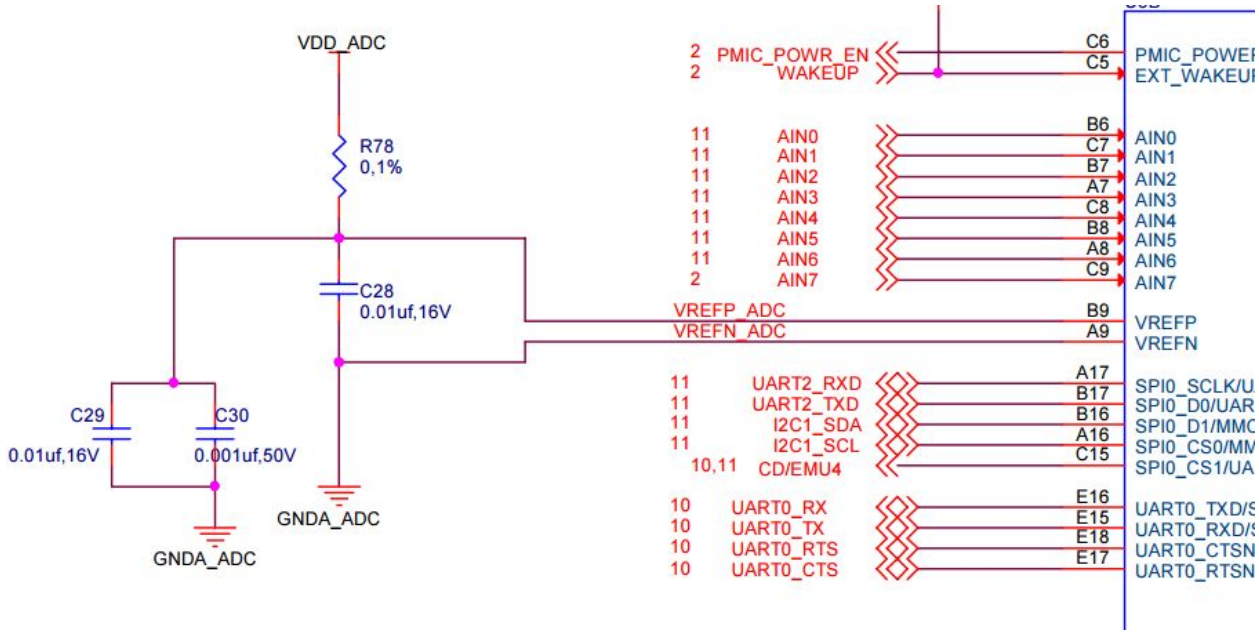


Figure 3: AM335 Analog Reference Voltage Structure

From Figure 2, we know that the IR sensors are capable of outputting voltages of over 3V, which could possibly damage the ADC of the AM335, so we needed to figure out a way to keep the input voltage to the ADC less than 1.8V. We first attempted to do this using a voltage divider with both legs set to 1kΩ. When viewing the output on the benchtop multimeter, this proved to be a good solution, but when reading the values on the Beaglebone analog input pins, we were unable to get a consistent reading. We changed resistor values multiple times and in the end were unable to get clean results using a voltage divider. We ended up implementing an op amp chopping circuit, see Figure 4, that allowed the signal to pass until it got to 1.8V at which point it held the value constant.

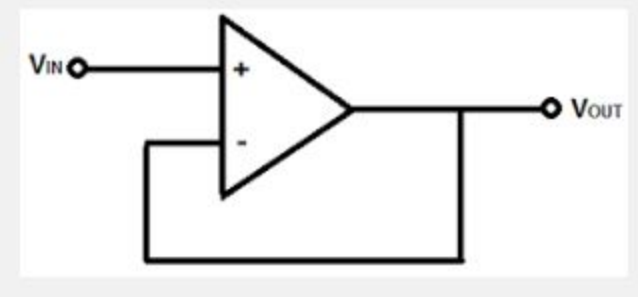


Figure 4: Op Amp Buffer Circuit

Using this circuit, we still were encountering erroneous values being read by the Beaglebone ADC and we found that there was a significant amount of noise and ripple on the outputs of the sensors. We were unable to determine the root cause of the noise, but we used a low pass filter to try and mitigate the effects as much as possible. We also used several decoupling capacitors at the input of the op amp power supply which helped reduce the noise as well.

The infrared sensors are read on a 100 ms timer interrupt and if the value is above or below a specified value, an interrupt is sent to the motor driver process to alert it of an impeding object or the removal of a previously impeding object. The reading of the analog inputs with the timer interrupt allows the rover to run uninterrupted as long as no objects impede its progress which makes for a more efficient process.

2.3 Joystick

To provide basic directional control of the tank we interfaced with an Adafruit Thumb Joystick breakout board. This allows us to provide basic forward/backward and turn-in-place capabilities. For our first implementation of the joystick, we stuck with a simplistic 4-way operational mode outlined below:

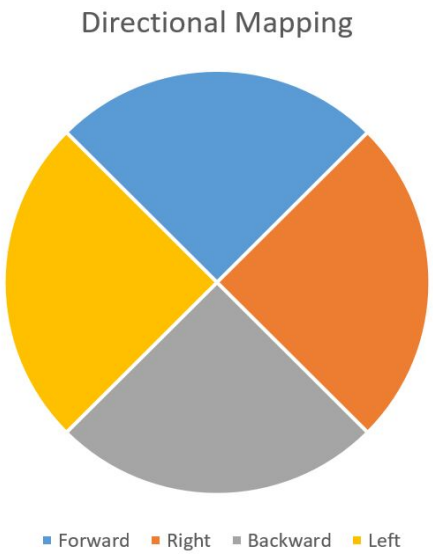


Figure 5: Directional mapping of joystick

Each slice of the joystick's operation maps to the following actions:

Direction	Action
Forward	Sets PWM duty cycle to 100% in forward direction
Backward	Sets PWM duty cycle to 100% in reverse direction

Right	Sets PWM for both motors to 100%, with left set to forward and right motor set to reverse
Left	Sets PWM for both motors to 100%, with right set to forward and left motor set to reverse

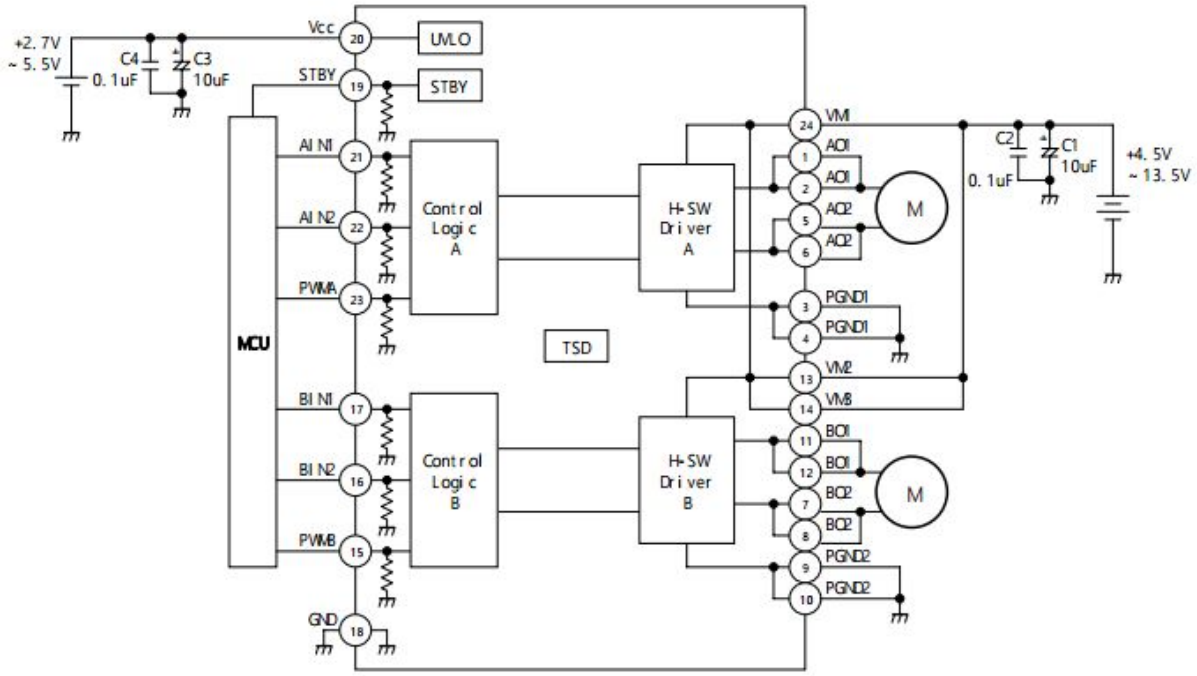
Our tank is therefore limited to very simple movements in the and is unable to make arcing turns. The logic behind this control is found in motor_control.c and can summarized as follows:

1. Define an event handler that is triggered when a sensor reads too close to an object and block movement in the direction corresponding to the sensor's placement (Ex. if the front sensor is triggered, block forward movement, but allow reverse). If the sensor drops back below a certain threshold, allow the corresponding movement again.
2. Read the X and Y data from ADCIN4 and ADCIN5.
3. Determine joystick's position based on figure 5.
4. Apply action based on the quadrant that joystick is in
5. Repeat

3.0 Conclusion

In conclusion, this lab was the first major step in the accumulation of the systems built for this class and has set the stage well for us to complete our final project, incorporating each of the components described throughout this report, and a few new ones. Implementing the motor controls and PWM early was a huge step towards tuning the motors and controls to get a smooth and controllable reaction from our rover for the final. However, almost even more importantly, the timer interrupts will be the basis of our functional controls for the “smart” system.There are a number of additional interrupts that we can implement to make out system robust. One of such is a watchdog timer interrupt that will protect our system from malfunctioning by watching critical components or potential bugs and executing a hard reset during operation if necessary. Such interrupts are critical in many embedded applications. Lastly, there are a number of small issues that we can clean up from this last project to make our final much more reliable. The noise from IR sensors was potentially interfering with the functionality of the interrupts. Because the ADC pins on the Beaglebone are all interconnected, over-voltage on one will lead to an overvoltage saturation on the others, so we need to eliminate any faulty sensors before we can count on precise interrupts without large reaction buffers or deadbands.

4.0 Appendix



Appendix A: TB6612FNG Application Diagram