Sneha Patri
Jamie Papa
Shyamal Anadkat

Lab 3: Remote Motor Control with Feedback

LAB GOALS

- Using the motor controller in our ECE 353 board, move the motor in our robot using receive and transmit data packets.
- Implement the encoders.c file based on the two encoders that each port has and decide which one is chosen.
- Use two GPIO pins as interrupts and create an interrupt handler in interrupts.c
- Based on the physical distance traveled by the robot and the pulse measured by the encoder find the number of pulses per inch. Also deliver the oscilloscope capture/readings for each of the encoder channels.

STEPS TAKEN TO ACCOMPLISH THE GOALS

- Follow the steps provided in the lab to set the IDs in the ecelab3.c file.
- Compiled the ECE-353-RobotController project to have the joystick control our robot.
- We followed steps 2.2 in the Lab pdf. rfInit() is called in the initializeBoard function
- In the main function add code to receive data from the 353 board controller and busy wait until data is received.
- We checked what data packets are being received based on the upper 16 bits of the 32 byte data received. Depending on whether it is “left”, “right”, and “forward” and “reverse” the upper 16 bits of the data being received changes, which will indicate the direction of movement to our motor. To actually move the motor we shall call the respective driver functions in the drv8833.c file.
- These PWM peripheral drivers were also used to control speed.
- Next step of the lab was to understand the function of encoders. There are 2 encoders in the entire robot, each encoder has a different output, one for left and one for right. We would choose which of these encoders will record the value of the pulse.
- Next we completed the encoderInit function in boardUtil.c file to initialize the encoders.
- Then the interrupt service routine was completed for the GPIO, to record the number of pulses for each of the encoder channels.
- We made the root run for 10 inches and calculated the number of pulses.
- We used the average of these values we calculated the pulses/inches.

- The subroutine in encoders computes the number of pulses required to accumulate for a given distance.

PROBLEMS FACED DURING IMPLEMENTATION

- It took us a while to understand the quadrature encoders. The left encoder was not sensing the pulses correctly.
- Also, we had to change the Launchpad due to the difference in pin headers (sometimes they did not attach firmly and caused unexpected behavior)

READY REFERENCE

- Ports F and C were used for the GPIO handler, based on the datasheet as the encoders were connected there.
- Controller ID: 51 for 315, 50 for 353
- PC5: Right A encoder, PC6: Right B Encoder, PF1: Left B Encoder, PF0: Left A encoder
- Wireless packet info: F, W + duty cycle for forward; R + V + duty cycle for reverse, R + T + duty cycle for right, L + F + duty cycle for left and S + T + duty cycle for stop [4 bytes of data]

IRQ COUNTS (rounded average):

IRQ Count Value for **10 inches: 138**

IRQ Count Value for **30 inches: 385**

IRQ Count Value for **100 inches: 1264**

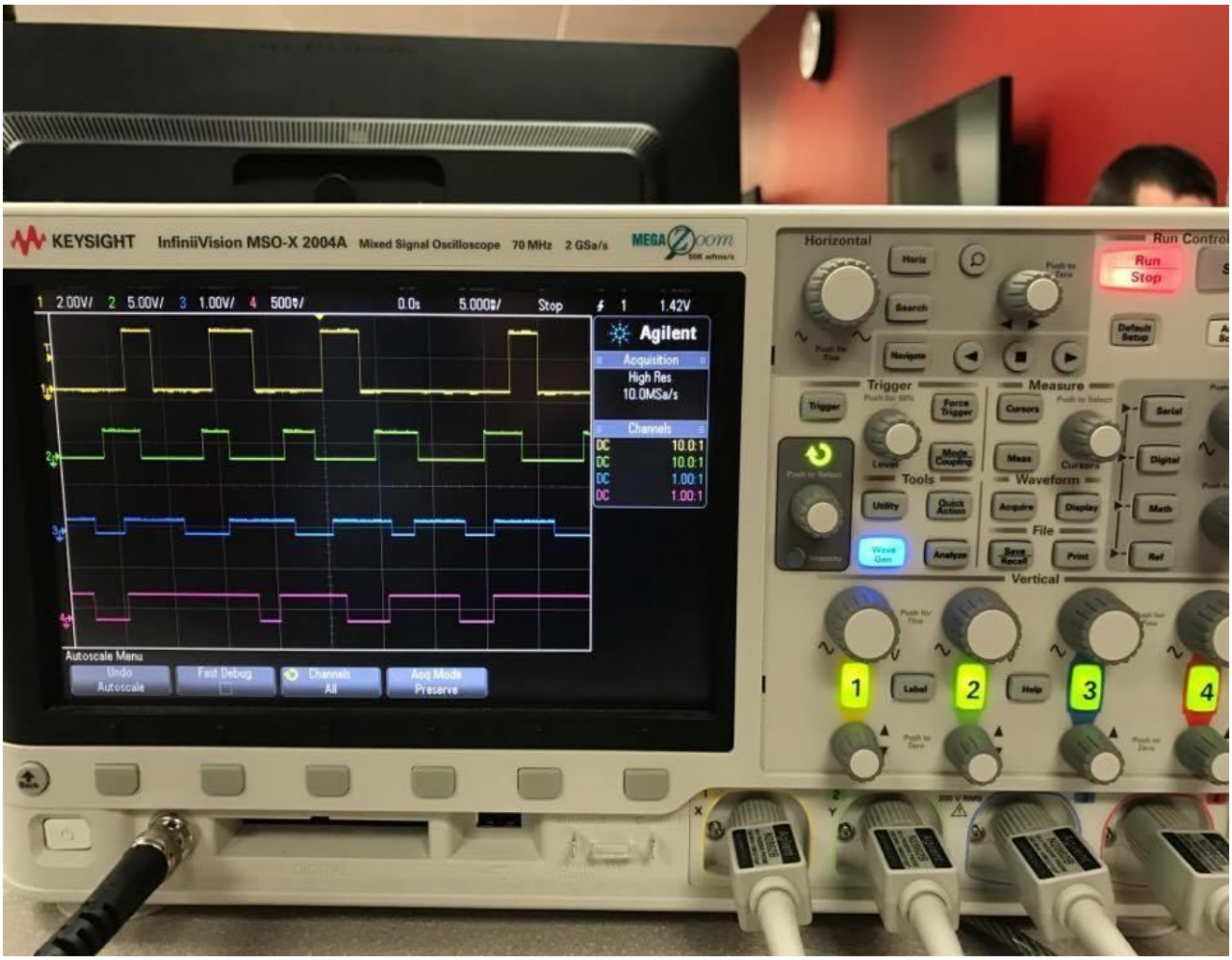

Figure 1:OSCILLOSCOPE CAPTURE for encoder pairs

KEY

>> Blue: PC5

>> Red: PC6

>> Yellow: PF1

>> Green: PF0

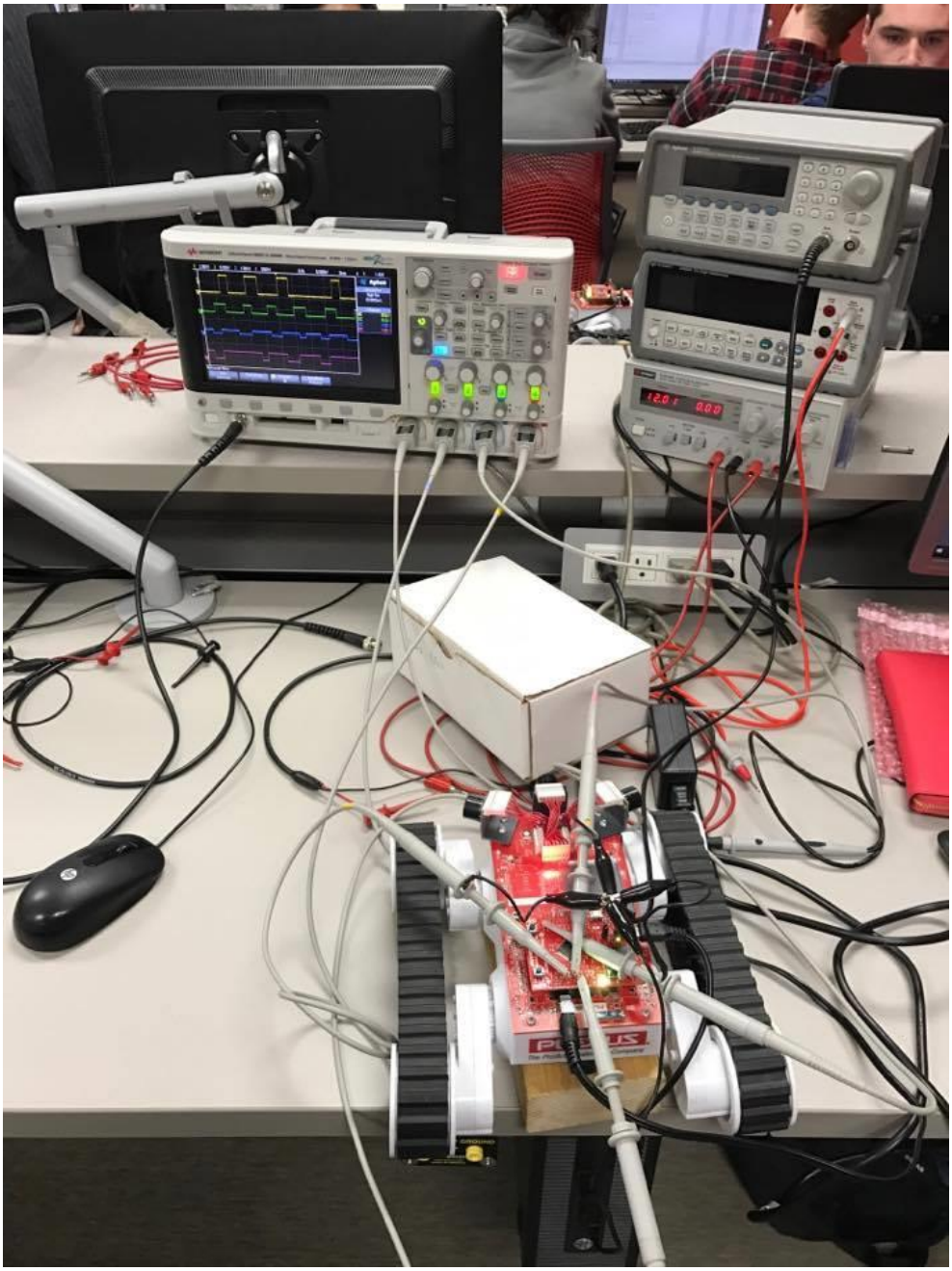

Figure 2: Additional images to demonstrate readings