

## Lab 6 CAN Network and Robot Interfaces

- Goals**
- Develop a layered communication system,
  - Design and implement a hardware/software interconnect protocol.
  - Interface two sensors and one motor needed for the robot.
  - Use communication skills to work effectively as team.

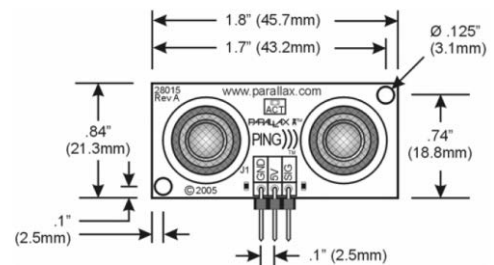
- Review**
- Textbook Chapter 8 and Sections 9.1 and 9.2
  - Chapter 17 of the TM4C123 data sheet explaining basic extended CAN,
  - IR distance sensor <http://www.ece.utexas.edu/~valvano/Datasheets/gp2y0a21yk.pdf>
  - **Ping)))** sensor <http://www.ece.utexas.edu/~valvano/Datasheets/PingAN.pdf>
  - **Ping)))** sensor <http://www.ece.utexas.edu/~valvano/Datasheets/PingDocs.pdf>
  - CAN reference: <http://www.kvaser.com/can/protocol/index.htm>

- Starter files**
- [http://www.ece.utexas.edu/~valvano/arm/CAN\\_4C123.zip](http://www.ece.utexas.edu/~valvano/arm/CAN_4C123.zip)
  - [http://www.ece.utexas.edu/~valvano/arm/PeriodMeasure\\_4C123.zip](http://www.ece.utexas.edu/~valvano/arm/PeriodMeasure_4C123.zip)
  - **Lab6.sch** circuit drawing with some parts used in Labs 6 and 7 (starter file),

*This lab must be performed in teams of 3 to 5 students, which will also be the Lab 7 robot competition teams. Approval of the team by the TA is required before the preparation is started. Only one version of the software need be developed for each team.*

### Background

Lab 6 has two or three sensors distributed across two computers. Each computer collects data from one or more sensor and shares the data with other computer using the CAN. For Lab 6 you will use the USB supply for +5V, but in Lab 7 the +5V will come from a 7805 linear regulator powered from the battery. The first sensor to interface is the **Ping)))**. This is an ultrasound transducer used to measure distance. You will use this sensor with input capture to measure the distance to a wall or to another robot. In Lab 6, you will interface one **Ping)))**, but your robot may have up to four **Ping)))** sensors.



The Sharp **GP2Y0A21YK** is an IR sensor, also used to measure distance. Similar to **Ping)))**, the goal is to measure the distance to a wall or to another robot. The sensor has an analog output inversely related to distance. The difficulty with **GP2Y0A21YK** sensors is that they are noisy. You must add both analog and digital filters in order to create a more stable and reliable measurement. Your system will also require some software calibration and calculations to implement a quantitative measure of distance. The Lab 7 robot will have up to four of these sensors, but only one will be used in Lab 6.

In Lab 6 you will have a third microcontroller separate from the two microcontrollers used in the CAN network and sensor interfaces. This third microcontroller will spin one DC motor in both directions in an open-loop fashion. In Lab 6 the DC motor should be powered from an 8.4V bench supply. This third microcontroller will send PWM signals to the motor driver circuit. In Lab 6 this third microcontroller should be able to adjust the duty cycle of the PWM output so the software can make the motor spin fast or slow. The motor driver system will be tested on a separate breadboard. In Lab 7, you will build two copies of this circuit on the robot protoboard and power it from the battery. However in Lab 6, you are just running one motor in open-loop fashion.



If there are 5 students on the team and you need a 3<sup>rd</sup> sensor, you could add a touch/contact sensor (a mechanical bumper switch). The data from all sensors is displayed on the LCD. The CAN interface must be used. Interrupts, FIFO queues, semaphores, and the OS must be used in an appropriate manner.

The hardware/software system is modular, because in Lab 7 you will create up to four copies of the IR distance sensor, up to four copies of the Ping))) , and two copies of the motor output driver. You will integrate these components on the two microcontrollers in Lab 7.

There are background tasks on each node: data acquisition and CAN I/O. The data acquisition threads will measure parameters (Ping, IR, or touch sensor) and send its measurement to a companion foreground thread (like Labs 2 and 3). A periodic task will start the **Ping)))** measurement every 100 ms. The time delay will be measured with an input capture interrupt. A periodic task will sample the ADC measuring distance with the **GP2Y0A21YK** sensor. Make sure the execution times of the background threads are short and bounded. A foreground thread will manage the CAN I/O and the LCD display. The CAN identifier will specify the data type. The data field can be formatted however you wish. The CAN receiver thread runs in the background, accepting messages from the other nodes. Since the CAN is dedicated, you can use CAN filters however you wish. As always, you are allowed to implement the system in an alternate manner, as long as the basic educational objectives of the lab are achieved. The appropriate use of semaphores and FIFO queues are expected.

Choose a CAN channel bandwidth between 500,000 and 1,000,000 bps, and leave it as a constant. Also leave the time-quantum settings as shown in the CAN starter files. To change data bandwidth, you will adjust the ADC sampling rate for the **GP2Y0A21YK** IR sensor. Start with a data acquisition/transmission rate so slow (e.g., 500/sec) that collisions will be rare, and all the FIFOs are usually empty. Then, keeping the CAN channel bandwidth fixed, increase the sampling rate (number of ADC samples/sec and number of CAN transmissions/sec) until the maximum bandwidth is reached, without loss of data. It is possible to implement this network using just the **Data Frame**, ignoring the **Remote Frame**, the **Error Frame**, and the **Overload Frame**. Similarly, you can implement just the **Standard CAN 2.0A**, instead of the **Extended CAN 2.0B**. CAN does force a priority structure on the network, but you can implement priority however you wish.

#### Preparation (do this before your lab period)

1) Download, unzip, and compile the CAN starter project. Integrate the low-level CAN functions into your Lab 3 or Lab 4 OS. Add semaphore calls to the CAN routines in appropriate places. Include any Lab 4 software you might wish to include in your robot. This application sends 4-byte messages. You may wish to increase the message length to 8 bytes.

2) Assign I/O pins for your Lab 7 robot. I recommend you use two PWM outputs on a TM4C123 board to control the motor. (Alternatively, e.g. if you have a microcontroller that does not have PWM capabilities, you can create PWM outputs in software using the timers.) Overall, you will need four input captures for the four **Ping)))**, and four ADC channels (for the four **GP2Y0A21YK** IR sensors), four digital outputs for the two DC motors two of which should be PWM, and some inputs for bumper switches or mode configurations. In addition, you should keep the UART interface for the interpreter, and the LCD to monitor robot status. The sensors and motors must be interfaced using two microcontrollers. You must be friendly and document your interfaces well. It is easier to design for the Lab 7 integrated sensor system now, than it will be to move a sensor from one pin to another later.

3A) One student in the group designs the interface for the **Ping)))**. There is one **SIG** pin used for both output and input. The fundamental approach will be to 1) make the **SIG** pin an output; 2) issue a 5  $\mu$ s output pulse (which causes a sound pulse to occur); 3) switch the **SIG** pin to back to an input; and 4) measure the time until the echo is received. The HCSR04 sensor is a little easier to interface because it has a separate input and output pin.  $t_{IN}$  will be a pulse width measurement using input capture. The pulse width time is equal to twice the distance to the object divided by the speed of sound. The distance  $d$  is therefore  $d=c*t_{IN}/2$ , where  $c$  is the speed of sound, and  $t_{IN}$  is the time for the sound to travel to the object, reflect and travel back to the sensor. You will perform this measurement about 10 times per second. You will need 5V to power the **L293**, **Ping)))** and **GP2Y0A21YK**. Use a five volt tolerance pin to interface the **Ping)))**. Not all pins are 5V tolerant. Draw the interface in PCBArtist (or another similar CAD program.) Using the appropriate OS commands, write foreground and background software that samples distance approximately 10 times a second and sends the data to other node on the system using CAN. You might want to disable interrupts during the 5 $\mu$ s pulse ( $t_{OUT}$ ) if you are using blind cycle counting for the time delay, so the OS does not suspend the thread during the pulse output.

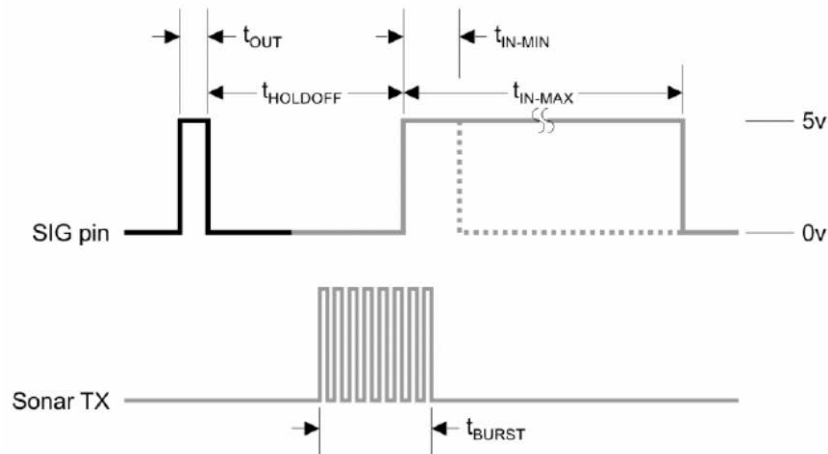


Figure 6.1. Ping))) sensor signals. See PingDocs.pdf for an explanation of this figure.

3B) A second student in the group designs the interface for one **GP2Y0A21YK** IR distance sensor. It, like the Ping))), is powered with +5V. Read the data sheet for the suggested power line capacitor (try using a bigger capacitor than suggested to see if it helps reduce the noise). The analog output signal has two types of noise. Notice the huge noise spikes in Figure 6.2 occurring about every one ms. On the spectrum analyzer you will also see both white noise and periodic EM field noise. During the procedure part of this lab you measure the noise of your system using a spectrum analyzer. The first step is to design an analog low pass filter with gain=1. If you power the rail to rail op amp with 3.3V, then the microcontroller ADC will be protected from voltages above 3.3 V or below 0V. The wheel diameter is about 8 cm, and the motors will spin up to 120 RPM, giving a maximum speed of about 50 cm/sec. Let  $x(t)$  be the distance to the wall as the robot travels at 50 cm/sec. Using the transfer function of the transducer, estimate the maximum slew rate,  $s$ , in volts/sec needed from the sensor interface. If the input were to be  $V=Asin(2\pi ft)$ , then the maximum slew rate  $dV/dt$  is  $2\pi fA$ . Assuming  $A$  to be about 1 V, if the maximum slew rate is  $s$ , the maximum frequency response needed is  $s/2\pi$ . Design an analog LPF filter with a cutoff frequency of about  $10*(s/2\pi)^{1/2}$ . Draw the interface in PCBArtist (or another similar CAD program.) Using the appropriate OS commands, write foreground and background software that samples distance approximately 2 times the LPF cutoff frequency and sends the data to the other computer on the system using CAN. Figure 6.3 plots data collected with the **GP2Y0A21YK** IR sensor employing a 2-pole 20 Hz Bessel LPF. Implement a 3-wide median digital filter to remove the noise spikes. Another nonlinear filter could be used in place of the median if you want.



Figure 6.2. GPY0A21YK sensor output, distance=20 cm, DC mode, voltage scale is 0.4 V and time scale is 1 ms.

<sup>1</sup> This cutoff should be around 10 to 20 Hz.

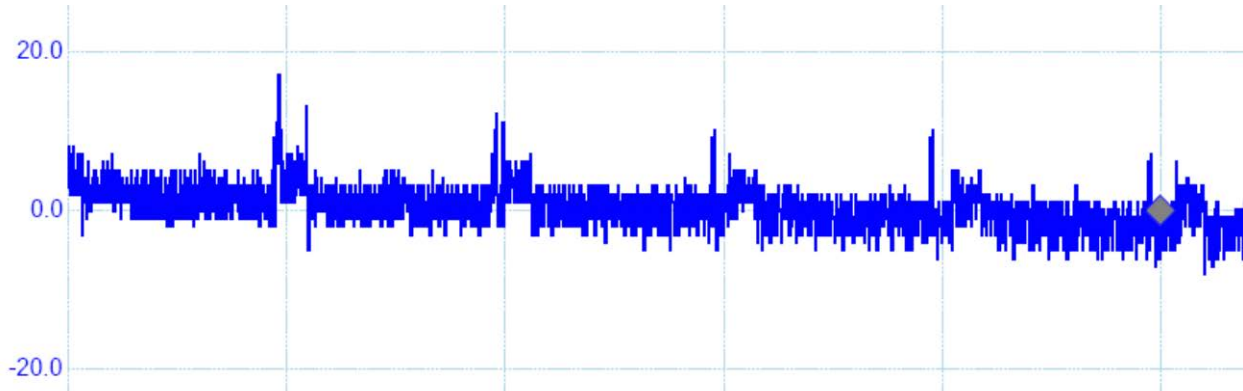


Figure 6.3 GPY0A21YK filtered output, distance=20 cm, AC mode, voltage scale is mV and time scale is 1 ms.

3C) A third student in the group designs the interface for one **DC motor**. The interface must allow for the software to control direction and power. Controlling direction will require an H-bridge, and controlling power will require PWM output from the microcontroller. This motor interface need not be integrated into the other components in Lab 6 (integration will occur in Lab 7).

<http://youtu.be/p4OevDMCuKo>

Figure 6.4. Measurements of motor voltage. The DC motor is controlled via PWM.

4) Finally, write background/foreground threads that accept CAN messages from the network. The foreground thread displays the most recent data from all nodes, including itself. Count the number of times a packet is be lost (i.e., the CAN receiving ISR calls **CanFifo\_Put** and the FIFO is full).

**Procedure (do this during your lab period)**

- 1) Modify your Lab 3 or Lab 4 OS to run on the two microcontrollers.
- 2) Test the CAN physical layer of the network. There are terminating resistors at each end of the cable. These resistors eliminate reflections and limit the rise/fall times of the voltages on the network. To test the network hardware, you can run one of the StellarisWare/Valvano starter examples. Next you will debug your integration of the CAN programs into the two computers.
- 3) One by one, debug the sensor interfaces. Please verify proper operation of the circuit before connecting to the microcontroller. Also remember to turn off both 3.3 and 5 V power when moving parts and wires on your circuit. Double check the power connections every time before applying power.
- 4A) Using known distances, calibrate the **Ping** so it measures distance accurately. The **maximum deviation** (or **span**) is the difference between maximum and minimum data points given a fixed distance. The **range** is the minimum and maximum distances at which the measurements can be accurately obtained. Write software that measures the average, standard deviation and maximum deviation of 10 consecutive measurements (one second). The walls of the race track will be pieces of 3.5 by 3.5 inch cedar wood. The floor will be the carpet in the 3<sup>rd</sup> floor lounge. The 22 cm wide robot will travel down a race track about 80 cm wide. Envision trying to speed your robot down the race track keeping a constant distance to the right wall. At what heights from the floor would it work? At what angles to the wall would it work? In order to determine the range, accuracy and noise level, take measurements (average, standard deviation and maximum deviation) at 5 known distances while detecting the race wall on the 3<sup>rd</sup> floor carpet creating a table like the one below. Calculate accuracy as the average absolute value of the differences between truth and measured. Standard deviation and maximum deviation are measures of noise.

Truth $d_T$	Measured $d_M$	Standard Deviation	Span

$$\text{Average accuracy (in cm)} = \frac{1}{5} \sum_{i=0}^4 |d_{Ti} - d_{Mi}|$$

4B) Using known distances, calibrate the **GP2Y0A21YK** sensor so it measures distance accurately. Write software that measures the average, standard deviation and maximum deviation of consecutive measurements over a one second time period. Look at the data sheet to choose the optimal orientation of the sensor to detect the wall while the robot is moving. Using the same scenario described in procedure 4A, at what heights from the floor would it work? At what angles to the wall would it work? In order to determine the range, accuracy and noise level, take measurements (average, standard deviation and maximum deviation) at 5 known distances while detecting the race wall on the 3<sup>rd</sup> floor carpet creating a table like the one above. Similar to Ping))), determine accuracy, range and noise for this sensor. Given a constant distance, measure the noise at the output of your LPF with a spectrum analyzer.

4C) Measure the current and voltage delivered to the motor under a no-load condition. Once the wheel is placed on the motor shaft and the robot placed on the ground, the current will increase 4 to 10 times this amount. You will not be able to increase the voltage above the 8.4V from the battery, so the design of the H-bridge will significantly affect the speed of your robot.

5) Put all the pieces together so data measured in each node is available in both computers. The file system is not needed for Lab 6. Measure a typical CAN frame with a dual trace scope showing both the output of the microcontroller and CANH on the actual bus.

6) Measure the maximum sustained bandwidth of the system, under the conditions that virtually no data is lost. Keep the CAN channel speed fixed at 500,000 (or 1,000,000) bps and increase the sampling rate of the **GP2Y0A21YK** sensor, while measuring the number of lost packets. "Sustained" means the measurement should occur over many seconds, so that the steady state behavior is measured, and not a start-up behavior measured when all the FIFOs are initially empty. Which factor limits the bandwidth? Consider factors such as the speed of the CAN channel, the speed of the ADC converter, execution speeds of various background threads, execution speed of foreground program, or the LCD display speed.

#### **Deliverables (exact components of the lab report)**

A) Objectives (1/2 page maximum)

B) Hardware Design

Detailed circuit diagrams of the three sensors (preparation 4)

C) Software Design (printout of these software components)

CAN module including its FIFO

Software to implement the sensor measurements

Software to implement the distributed data acquisition

D) Measurement Data

Table of **Ping**))) data at 5 known distances and average accuracy (procedure 4A)

Table of **GP2Y0A21YK** data at 5 known distances and average accuracy (procedure 4B)

Spectrum of **GP2Y0A21YK** noise (procedure 4B)

Table of **QRB1134** data at 5 known distances and average accuracy (procedure 4C)

Scope traces of CAN signals measured on both sides of the SN65HVD1050 (procedure 5)

Measure the network bandwidth (procedure 6)

E) Analysis and Discussion (2 page maximum). In particular, answer these questions

1) What is one advantage of the **Ping**))) sensor over the **GP2Y0A21YK** sensor?

2) What is one advantage of the **GP2Y0A21YK** sensor over the **Ping**))) sensor?

3) Describe the noise of the **GP2Y0A21YK** when measured with a spectrum analyzer.

4) Why did you choose the digital filters for your sensors? What is the time constant for this filter? I.e., if there is a step change in input, how long until your output changes to at least 1/e of the final value?

5) Present an alternative design for your H-bridge and describe how your H-bridge is better or worse?

6) Give the single-most important factor in determining the maximum bandwidth on this distributed system.

Give the second-most important factor. Justify your answers.

F) Post-mortem concerning team member interactions (attached to the report)

- 1) Each team member evaluates each other team member including oneself  
Simply list one or two weaknesses.  
Simply list two or three strength characteristics.
- 2) Major failures in the way the team interacted (if any)
- 3) Major successes in the way the team interacted

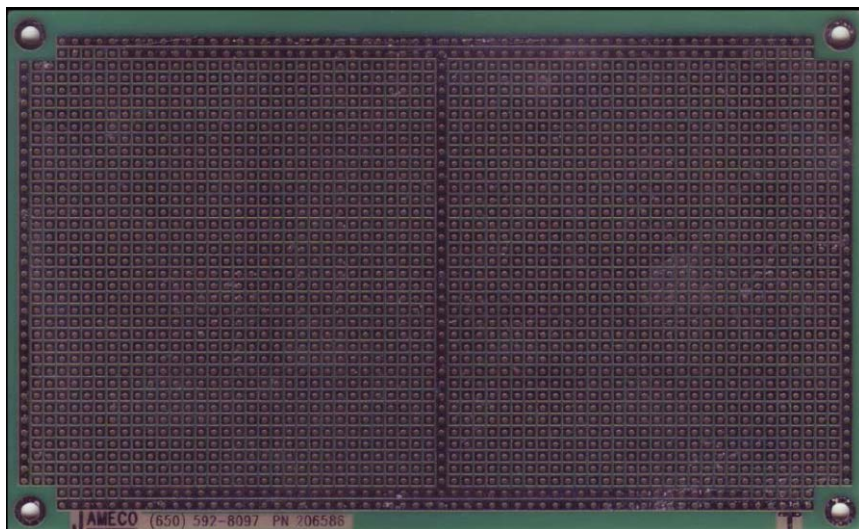
G) Peer Review (each student submits independently and confidentially directly to the TA)

Classify each team member including oneself as:

- worked harder than average (explain), worked an average amount, worked less than average (explain)

**Checkout (show this to the TA)**

Show the three sensors in operation creating scope tracings similar to Figures 6.1 to 6.5. Be prepared to discuss factors such as accuracy, noise, and reproducibility. Connect a logic analyzer to CAN signals PE4 and PE5. Explain these two signals as data are communicated across the network.



All soldering should be on this side

All components should be on the other side

Use solid 22 or 24 gauge wire with a 0.15in striped end, push in from other side and bend over.

Plan the system before soldering