Lab 9: Self-Driving Kart

Submission Due Dates:

Demo: 2022/12/27 17:20
Source Code: 2022/12/27 18:30
Report: 2022/12/29 23:59

Objective

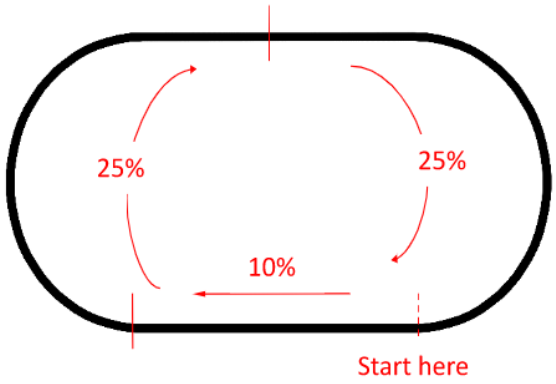
1. Get familiar with Pmod connectors for external devices
2. Get familiar with the 3-way track sensor and ultrasonic sensor
3. Learning to control typical DC motors via PWM signals

Description

In this lab, you will design and implement a homemade self-driving vehicle that follows a black-line track and stops when an obstacle is in front. Look into the template codes (lab9_top.v, motor.v, tracker_sensor.v, and sonic.v). Note that there are TODO items in these files. TAs will ask questions about how they work. You may also modify the template code if you wish to, but you have to be able to explain your changes.

Detailed Specifications

1. Regular track: (60%)
   The car should be able to navigate around the racing track autonomously. The racing track is defined by black electrical tape (with a width of 20mm) on a white background. Your car must find its way back to the racing track if it accidentally leaves the track. The car is determined as having left the track when no part of the car is touching the black line anymore. In addition, it must return to the last on-track position or before where it left off and continue the racing. Corner cutting is not allowed; that is, if your car departs from the track and skips to another point ahead of it in the curve, the run is invalid. The regular racing track and the grading policy are as follows (10% for the straight line; 25% for each curve):



2. Obstacles: (40%)
   The TA will place an obstacle on the track at random spots (at least 30cm ahead of the car). The car must come to a complete stop before hitting the obstacle, then continue after the obstacle is removed. The obstacle will be larger than the car. For each successful avoidance, you get 20% (10% for stopping properly, 10% for resuming the movement). However, you will get 0% if the car hits the obstacle, even if

it keeps moving after removing the obstacle.

3. Bonus track: (10%)

   There will be a hidden racing track. The track will be unveiled at the demo time and is only eligible for those who pass the regular track. The car must be able to complete the track with obstacle avoidance within 3 minutes. There are several hints for the hidden track:

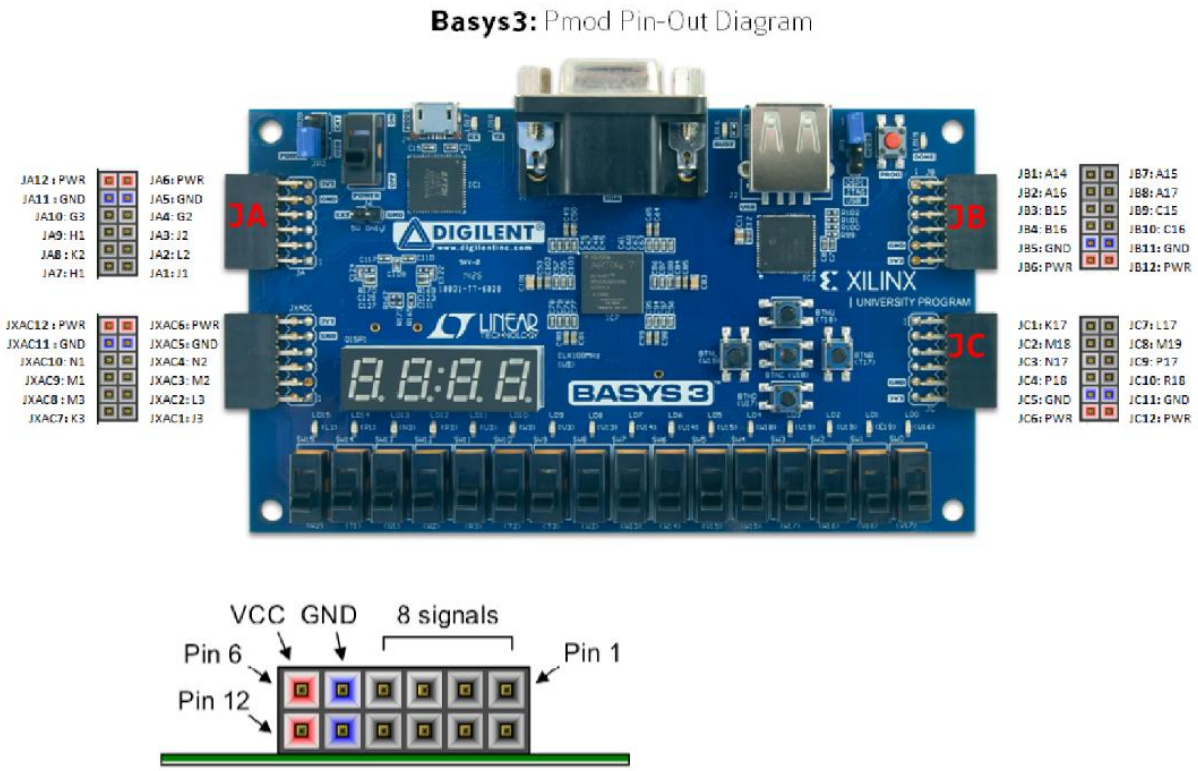
   - The track will not overlap itself
   - There are no dead ends
   - There are no right angles; all turns are curved

   Grading policy:

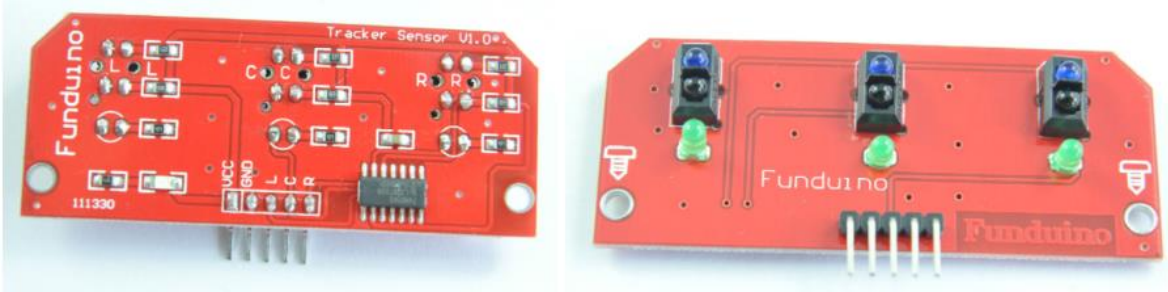
   - (5%) Complete the track either clockwise or counterclockwise.
   - (3%) Complete the track both clockwise and counterclockwise.
   - (2%) Avoid obstacles. This is only given if the track can be completed.

Hardware Details

1. Here is the pin-out diagram for the PMOD connector for reference:



2. 3-way track sensor

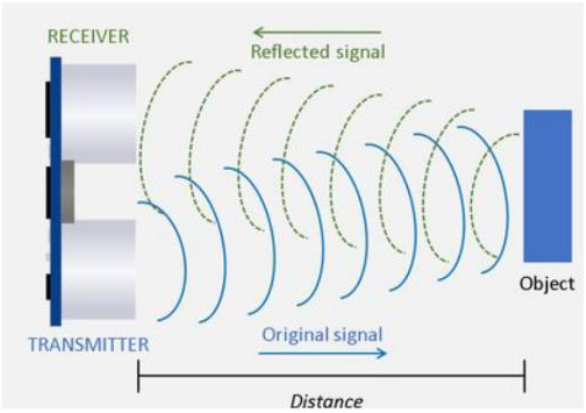


The sensor has three independent infrared (IR) sensors. Each IR sensor has an IR blaster and an IR receiver. If the IR is being reflected by the floor, the sensor will output HIGH. A white floor reflects IR, therefore, the sensor outputs HIGH. A black floor absorbs IR, and the sensor outputs LOW.

Pin connections:

- VCC pin: connects to the supply power provided by the FPGA board.
- GND pin: connects to the ground of the FPGA board.
- L pin: outputs the status of the left IR sensor. You should connect it to an input port.
- C pin: outputs the status of the center IR sensor. You should connect it to an input port.
- R pin: outputs the status of the right IR sensor. You should connect it to an input port.

3. Ultrasonic sensor:

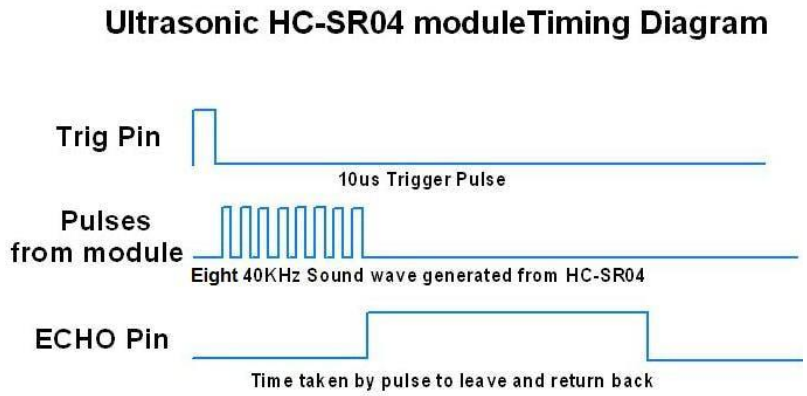


The HC-SR04 Ultrasonic sensor measures the distance between the sensor module and the object in front of it. The process is as follows (you should also trace the code in **sonic.v** to see how this is implemented and see how to interface with the sensor):

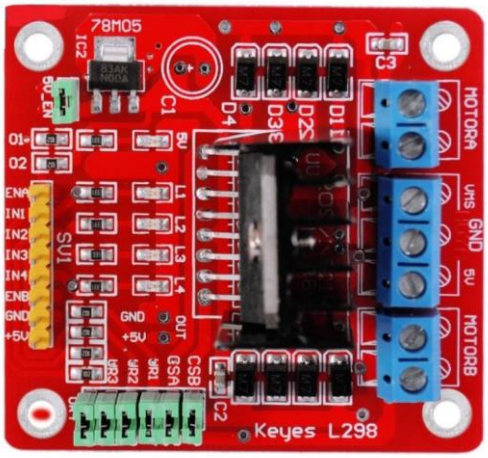
- First, send a 10us pulse to the “Trig” pin to trigger the sensor.
- The sensor then generates 8 pulses of ultrasonic soundwaves and determines the distance.
- After that, the “Echo” pin will output a long pulse. The length of the pulse is equal to the total travel time of the soundwave.
- Calculate the distance, that is, (pulse_length / 2) * 340(m/s).
- Each measurement should have a >60ms interval for better accuracy.

Pin connections:

- VCC pin connects to the supply power provided by the FPGA board.
- GND pin connects to the ground of the FPGA board.
- Trig pin triggers the sensor. You should connect it to an output port.
- Echo pin is the output by the sensor. You should connect it to an input port to measure pulse length.



4. L298N motor driver and motors

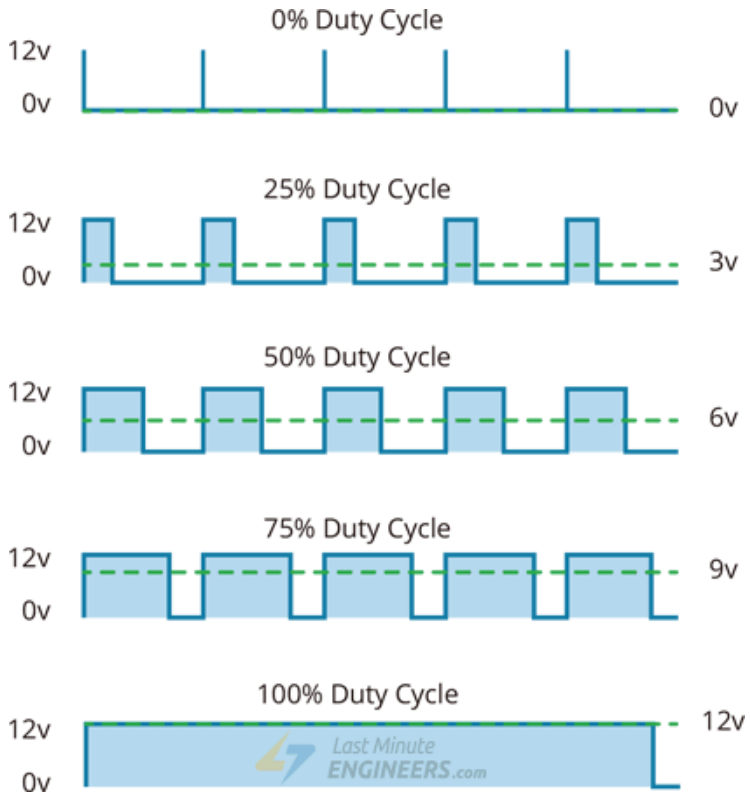


The L298N board is a dual H-Bridge motor driver which supports the speed and direction control of two DC motors at the same time. ENA, IN1, and IN2 control the motor A, while ENB, IN3, and IN4 control the motor B. You may use the sample **motor.v** file to interface with the driver.

- Direction controls:

IN1 (IN3)	IN2 (IN4)	Spinning Direction
LOW	LOW	Motor off
HIGH	LOW	Forward
LOW	HIGH	Backward
HIGH	HIGH	Motor off

- Speed controls:
  We send PWM signals to ENA and ENB to control the speed. A larger duty cycle results in a faster speed. Note that if the duty cycle is too small, there might not be enough power to drive the car forward. A cycle of 60~80% is recommended as a starting point.
- Additional notes:
  There may be slight speed differences between the two brushed DC motors used to power the car. These differences will add up and eventually make the car veer to one side even when going straight. You might need to compensate for this in your implementation.



Pin connections: (the yellow connections on the left)

- ENA connects to an output pin of the FPGA board. It controls the speed of motor A.
- IN1 and IN2 connect to the output pins of the FPGA board. They control the direction of motor A.
- IN3 and IN4 connect to the output pins of the FPGA board. They control the direction of motor B.
- ENB connects to an output pin of the FPGA board. It controls the speed of motor B.
- GND connects to the ground of the FPGA board
- 5V provides 5V power to the FPGA board. Connect it to the external power pins on the FPGA board.

Wire terminals: (the blue ones with screws on the right)

- MOTORA connects to a pair of positive and negative motor wires of motor A.
- VMS is the power input. Connects to the **positive** terminal of the battery.
- GND is the power ground. Connects to the **negative** terminal of the battery.
- 5V outputs the 5V power. Not used in this lab.
- MOTORB connects to a pair of positive and negative motor wires of motor B.

I/O Signal Specification

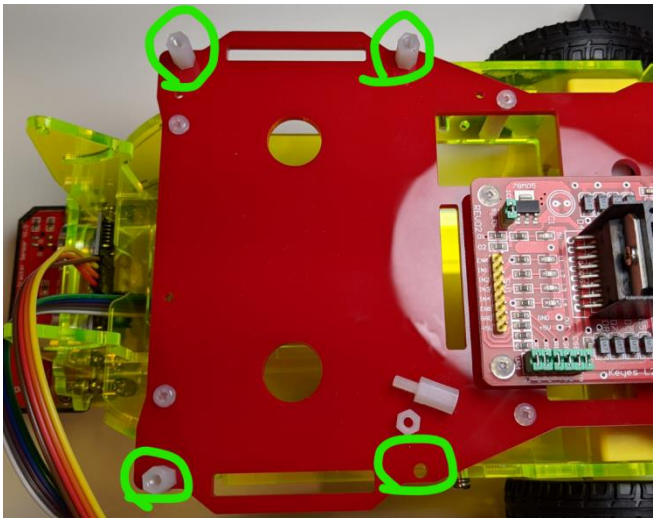
Here we provide the I/O connections defined in **lab9_constrains.xdc**. You may change the PMOD connections if necessary.

Name	Pin	Description
clk	W5	clock signal with the frequency of 100MHz
rst	btnC (U18)	active-high reset
IN1	JA1 (J1)	connected to “IN1” pin of the motor driver
IN2	JA2 (L2)	connected to “IN2” pin of the motor driver

IN3	JA3 (J2)	connected to “IN3” pin of the motor driver
IN4	JA4 (G2)	connected to “IN4” pin of the motor driver
left_pwm	JA7 (H1)	connected to “ENB” pin of the motor driver
right_pwm	JA8 (K2)	connected to “ENA” pin of the motor driver
trig	JB3 (B15)	connected to “trig” pin of the ultrasonic sensor
echo	JB4 (B16)	connected to “echo” pin of the ultrasonic sensor
left_track	JB8 (A17)	connected to “R” pin of the 3-way track sensor
mid_track	JB9 (C15)	connected to “C” pin of the 3-way track sensor
right_track	JB10 (C16)	connected to “L” pin of the 3-way track sensor

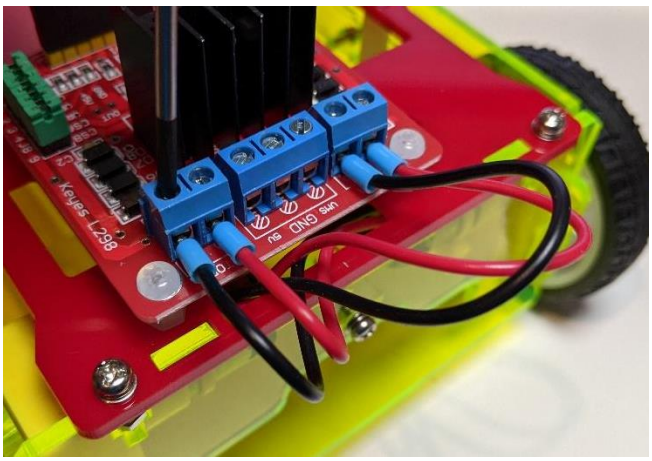
Assembly Instructions

1. Remove four plastic pillars on the car.

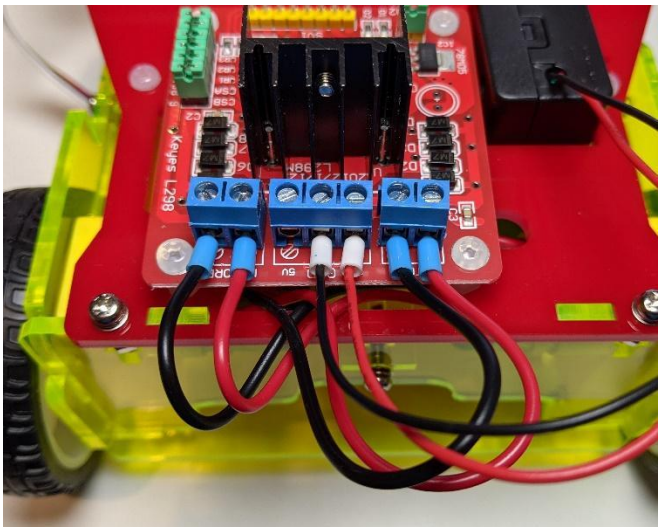


Tip: Put the washer back so you won’t lose it.

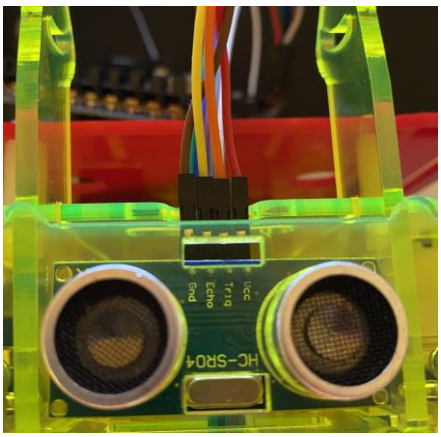
2. Connect the motor wires to the blue wire terminals. Insert the wire and use a flat-head screwdriver to tighten the terminals. Try to pull the wires gently to ensure they are secure. The polarity of the motor may not matter since we can easily change their spinning direction in the Verilog code.

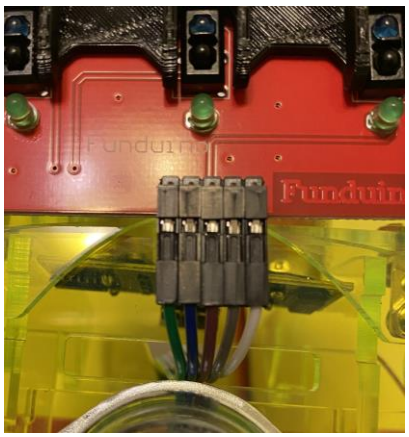


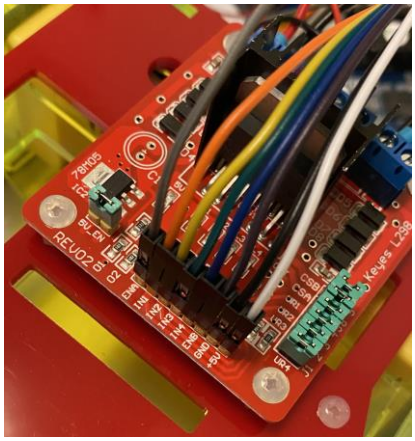
3. Connect the battery wires to the wire terminals in the middle. **The positive (red) wire to VMS, and the negative (black) wire to GND**. **Caution:** you WILL damage the hardware if you connect them incorrectly.



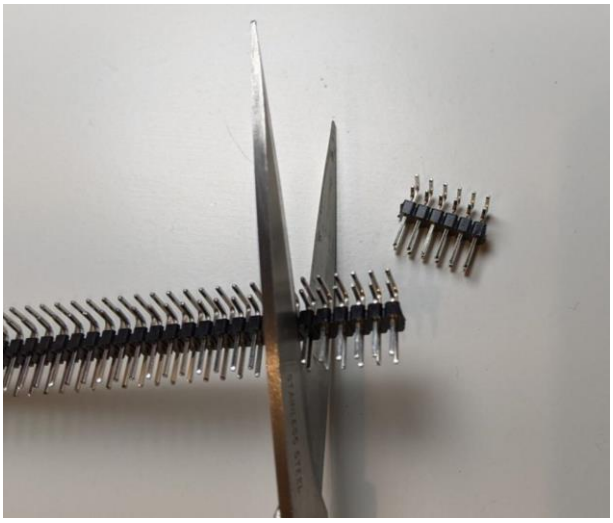
4. Connect jumper wires for the ultrasonic sensor, the 3-way track sensor, and the motor driver.
   Note: the jumper wires in each box set may have different colors from the ones in the photos below. Connect them with caution.



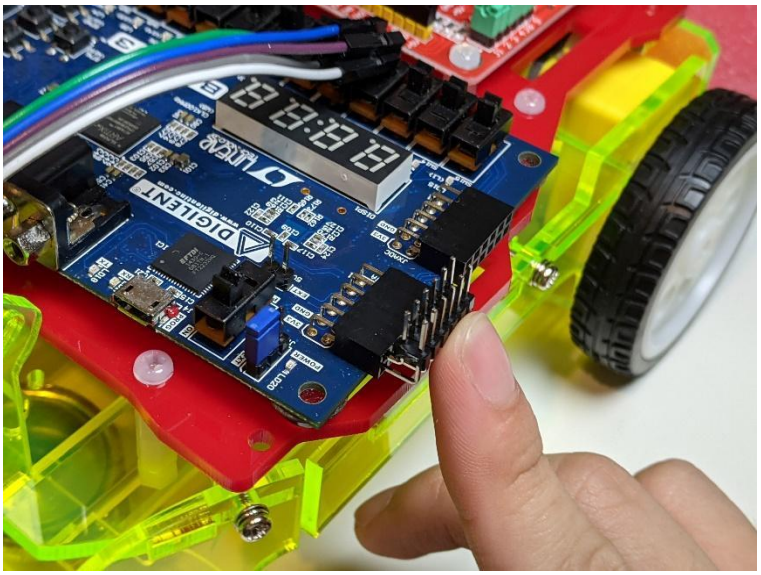




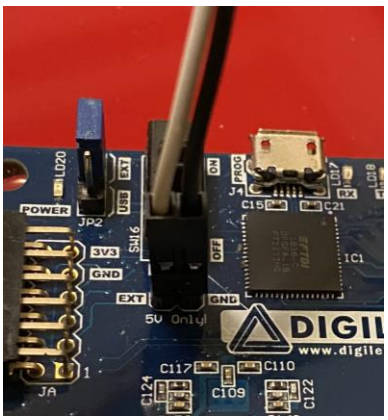
5. Cut 2 groups of six right-angle connectors, if they are not cut already.

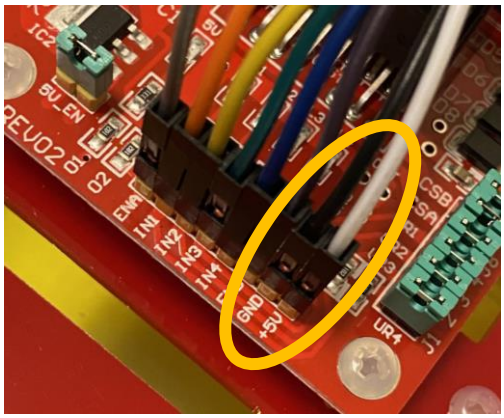


6. Place the FPGA board on top and insert the right angle pins into the Pmod connectors.



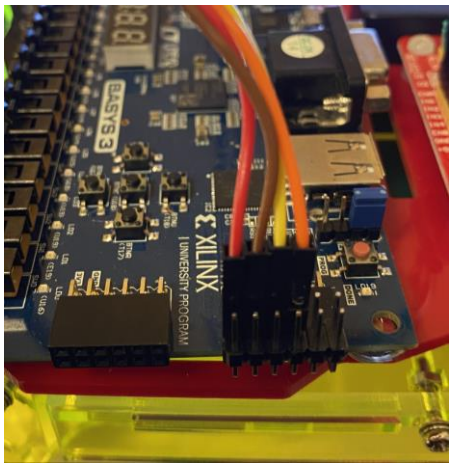
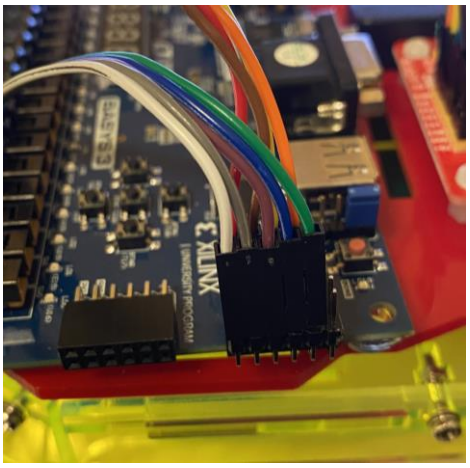
7. Move the power jumper to “EXT” to receive the supply power from the external power pins. Then connect the jumper wires from the motor driver’s power output to the FPGA. **Connect 5V to EXT, GND to GND**. **Caution:** you WILL damage your FPGA board if you connect the wrong way.





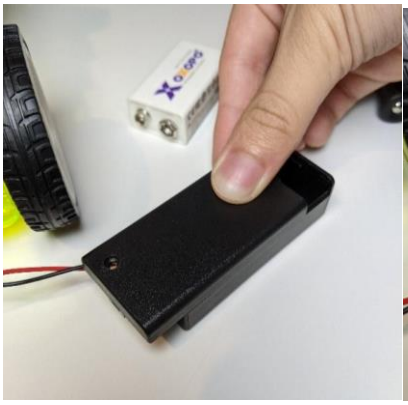
8. Connect the rest of the jumper wires to the PMOD connectors according to the I/O Signal Specification (or your own constraints file). **Be careful with the power connections** (3V3 and GND on the FPGA board). **You need to provide power (connect 3V3 to VCC and GND to GND) to the ultrasonic sensor**

and the 3-way track sensor. Caution: you WILL damage your FPGA board and/or your sensors if you connect the wrong way.


9. Insert the battery. **Be careful with the polarity**. **Be sure to double-check all the power-related connections before you turn on the power.** Now flip on the power switch, and you are ready to download your bitstream to the FPGA board!

Note: If you short-circuit or overload the battery, it may enter an over-current protection state. This may also happen if your duty cycle is too big. Connect the battery to a charger to restore it.



Questions and Discussion

Please answer the following questions in your report:

A. How does the PMW_gen module generate the pulse for the motors? Please explain the implementation in the PWM_gen module. What will happen if the duty cycle is extremely high or extremely low? Will that affect the car's performance on the track (e.g., speed, smoothness, or anything else)? If yes, please explain how.

B. How does the state transition in the PosCounter module work? Draw a state diagram and explain how it works. You should describe the concept of events instead of listing the signals.

Attention

- You may adjust your design and its parameters during the demo.

- You can add extra features to aid debugging and tweaking. For example, you can show the distance on the 7-segment display and show the status of the 3-way track sensor on three LEDs.
- You should hand in only one Verilog file, **lab9.v**. If you have several modules in your design, integrate them in lab9.v. **Please merge the modules in sonic.v, motor.v, and tracker_sensor.v into lab9.v.** Do not include debounce, one-pulse, and clock divider modules in the file.
- Do not hand in any compressed files, which will be considered an incorrect format.
- Please make sure your car can move without being connected to your PC by any cords. Consider using USB or SPI Flash programming modes.
- DO NOT copy-and-paste code segments from the PDF materials. Occasionally, it will also paste invisible non-ASCII characters and lead to hard-to-debug syntax errors.
- You should also hand in your report such as **`lab9_report_group01.pdf`** (where **`01`** is your two-digit group ID).
  - This lab is scored by groups.
  - You must list the group members and describe your job partition clearly in the report.
- You should be able to answer questions about this lab from TA during the demo.
- Feel free to ask questions about the specification on the EECLASS forum.