Lab 6 Advanced: Autonomous Kart

Submission Due Dates:

Demo:	2024/12/3 17:20
Source Code:	2024/12/3 18:30
Report:	2024/12/8 23:59

Objective

1. Familiarize yourself with the Pmod connectors for interfacing with external devices
2. Understand the functionality and usage of the 3-way track sensor and ultrasonic sensor
3. Learn how to control standard DC motors via PWM signals

Description

In this lab, you will design and implement a homemade autonomous vehicle capable of following a black-line track and stopping when an obstacle is in its path.

- Review the provided template codes: lab6_advanced.v, motor.v, tracker_sensor.v, and sonic.v.
- Pay attention to the TODO items within these files, as they require your attention.
- Be prepared to explain how the template code works; TAs may ask questions about its functionality.
- You may modify the template code as needed, but you must be able to justify and explain any changes you make.

Detailed Specifications

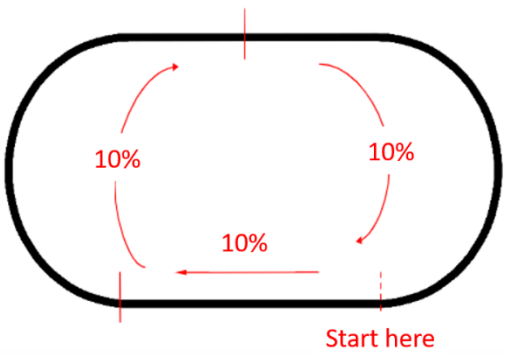
Baseline (70%)

1. Track: (60%)
   The autonomous vehicle must navigate the racing track independently. The racing track is defined by black electrical tape (20mm wide) on a white background. Your vehicle should fulfill the following requirements:
   - The kart must follow the track autonomously. If it leaves the track (i.e., no part of the kart overlaps the black line), it must find its way.
   - Upon leaving the track, the kart must return to the last on-track position or the point before where it left off and resume racing. Simply skipping ahead on the curve is not allowed.
   - The kart must not bypass sections of the track or cut corners. If it departs the track and skips to a forward point without properly reentering, the run is considered invalid.
   - A standard racing track will be provided, and the grading policy is detailed in the following section.

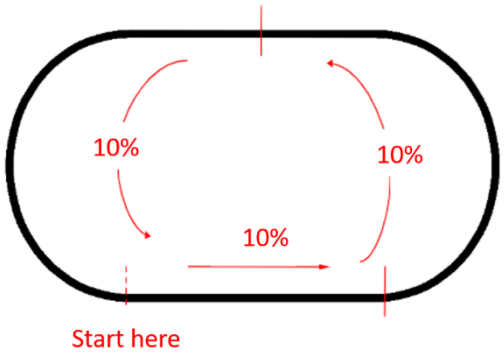
   Grading Policy:

   (1) (30%) Complete the track clockwise. (10% for the first straight line; 10% for each curve)

   (2) (30%) Complete the track counterclockwise. (10% for the first straight line; 10% for each curve)



Testcase: clockwise



Testcase: counterclockwise

The fastest group to complete the track will get a 10% bonus.

2. Obstacles: (10%)
   The TA will place an obstacle on the track at random locations, at least 30cm ahead of the kart. The vehicle must come to a complete stop before hitting the obstacle, and then continue along the track once the obstacle is removed. The obstacle will be larger than the kart. For each successful avoidance, you get 10% (5% for stopping properly, 5% for resuming the movement after the obstacle is removed). However, you will get 0% if the vehicle hits the obstacle, even if it resumes moving after the obstacle is removed.

Challenge (30%)

There will be a hidden racing track. The track will be unveiled during the demo time and is only eligible for those who successfully complete the Baseline 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 be a continuous loop without overlap.
- There is no dead end.
- Turns are curved, with no right angles.
- Random obstacles will be placed along the track. Hitting an obstacle results in 0% for the run.

Grading policy:

- (15%) Complete the track clockwise.
- (15%) Complete the track counterclockwise.

The fastest group to complete the track will get a 10% bonus.

Hint:

- Although the track is narrow, handle cases where two of the three IR detectors above the track detect the line (e.g., black/black/white or white/black/black). Ensure your navigation policy handles these scenarios effectively.
- Address edge cases where the sensor doesn't detect the line at all. Develop a recovery strategy for these situations.

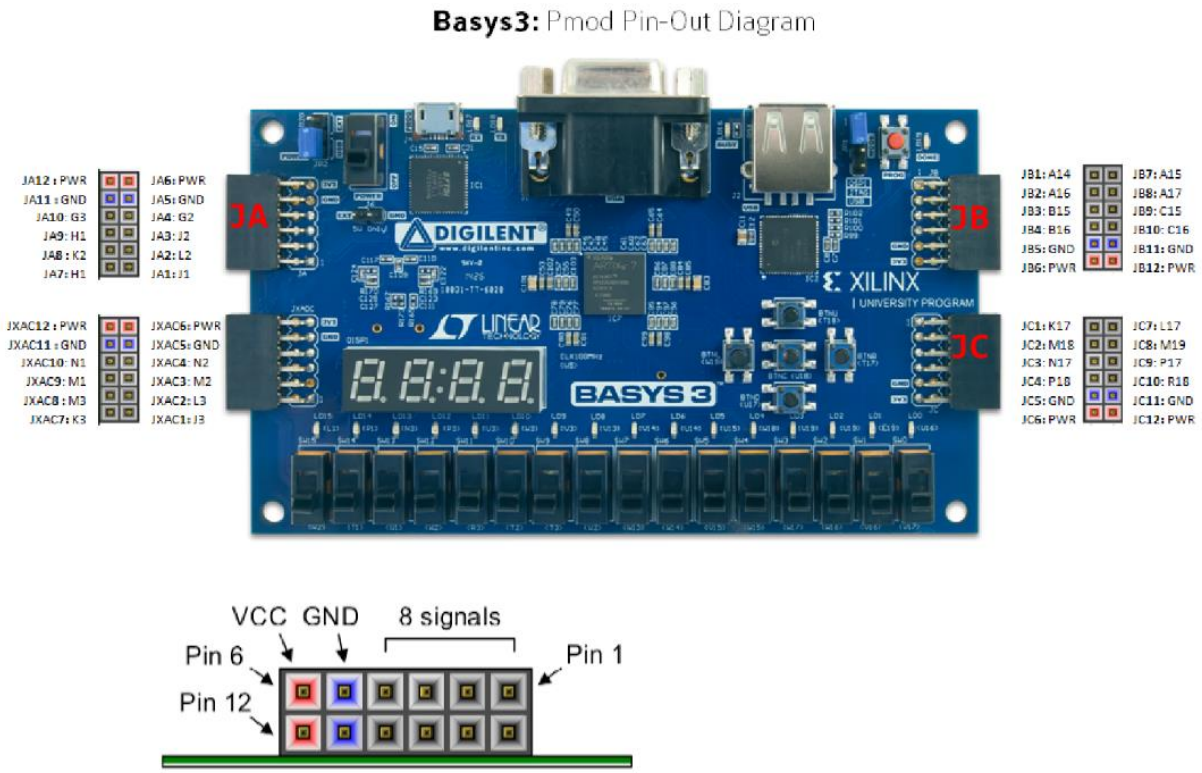
Note: Credits in the Challenge section will only be awarded if you can clearly explain your car's navigation policy to the TA.

Notice: Plagiarism or cheating will result in a grade of -100. The TAs will thoroughly compare your code with all submissions from this and previous years.

Hardware Details

In this tutorial, we will guide you through the assembly and control of the car.

1. Refer to the pin-out diagram for the PMOD connector for reference:



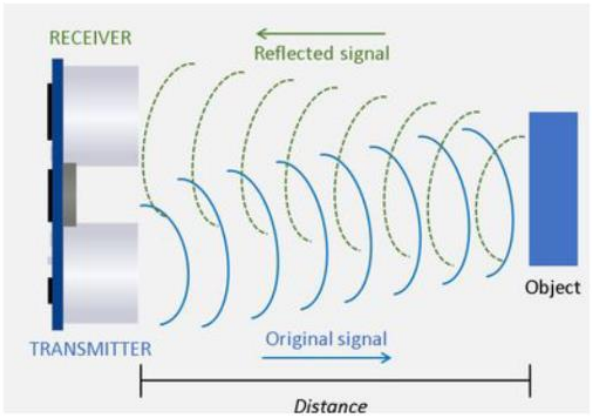
2. 3-way track sensor

The sensor has three independent infrared (IR) sensors, each with an IR blaster and an IR receiver. If the floor reflects IR, the sensor will output HIGH. A white floor reflects IR, so the sensor outputs HIGH. A black floor absorbs IR, so 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:

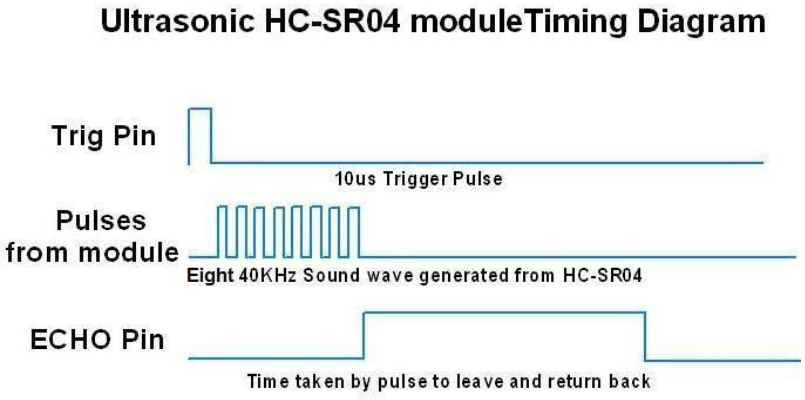


The HC-SR04 Ultrasonic sensor measures the distance between the sensor module and the object in front of it. The process is as follows (you should also trace the code in **sonic.v** to see how this is implemented and how to interface with the sensor):

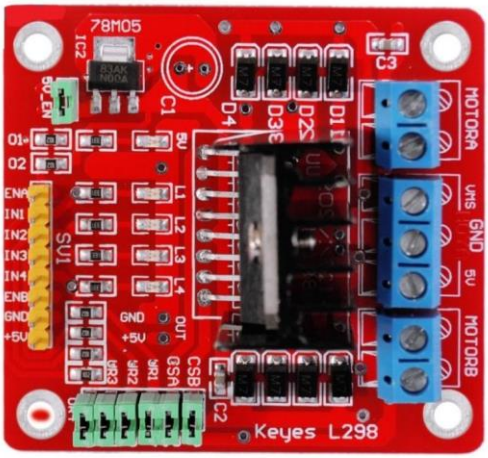
- First, send a 10 us pulse to the “Trig” pin to trigger the sensor.
- The sensor then generates 8 pulses of ultrasonic soundwaves and determines the distance.
- After that, the “Echo” pin will output a long pulse. The length of the pulse is equal to the total travel time of the soundwave.
- Calculate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

Pin connections:

- VCC pin connects to the power supply 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 sensor’s output. You should connect it to an input port to measure pulse length.



4. L298N motor driver and motors

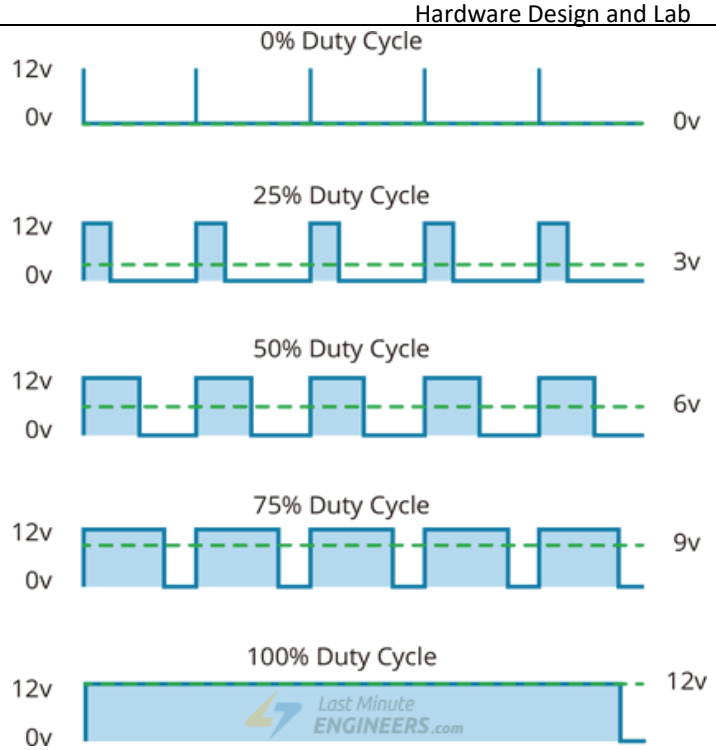


The L298N board is a dual H-Bridge motor driver that controls the speed and direction of two DC motors simultaneously. ENA, IN1, and IN2 control the motor A, while ENB, IN3, and IN4 control the motor B. You may use the sample **motor.v** file to interface with the driver.

- 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 higher speed. Note that if the duty cycle is too small, there may not be enough power to drive the car forward. A duty cycle of 60 to 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 for motor A.
- VMS is the power input and connects to the **positive** terminal of the battery.
- GND is the power ground and connects to the **negative** terminal of the battery.
- 5V outputs the 5V power but is not used in this lab.
- MOTORB connects to a pair of positive and negative motor wires for motor B.

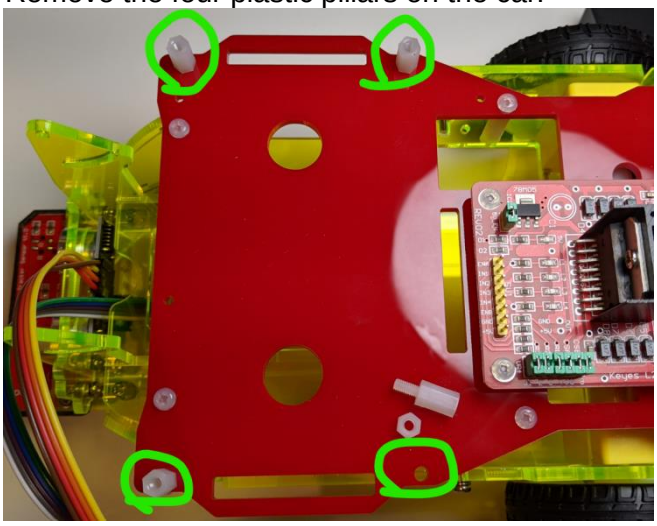
I/O Signal Specification

Here are the I/O connections defined in **lab6_advanced.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
right_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
left_track	JB10 (C16)	connected to “L” pin of the 3-way track sensor

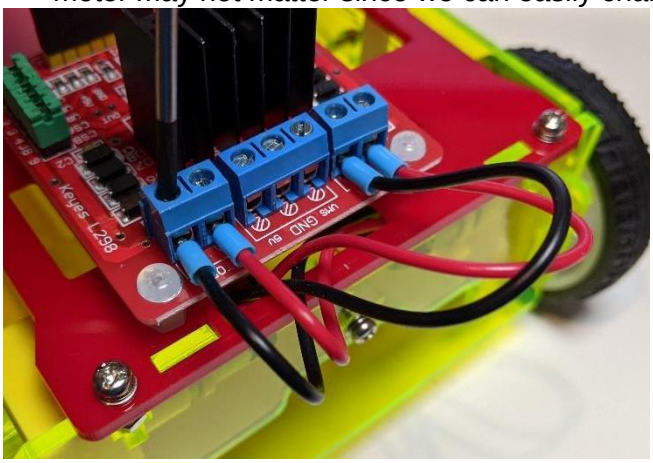
Assembly Instructions

1. Remove the four plastic pillars on the car.

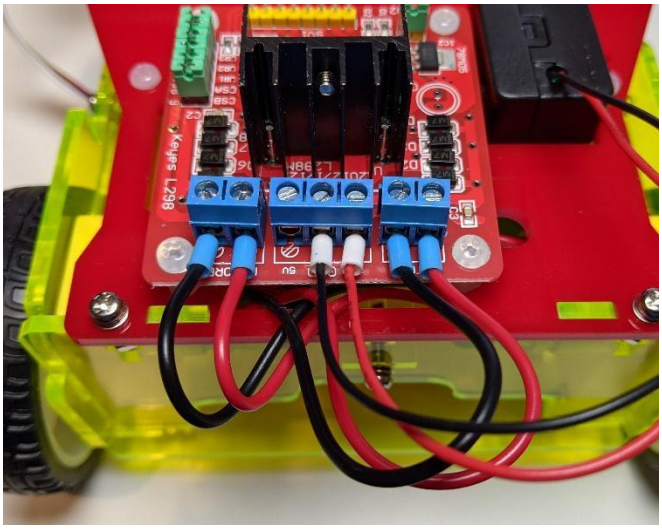


Tip: Put the washer back so you won't lose it.

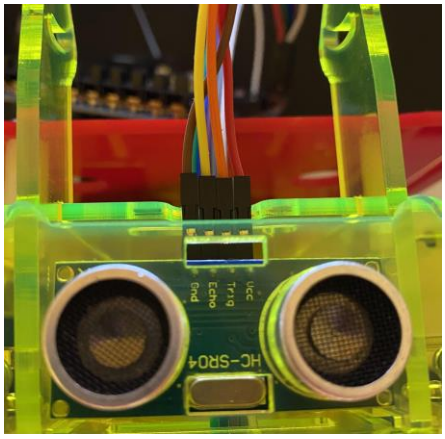
2. Connect the motor wires to the blue wire terminals. Insert the wire and use a flat-head screwdriver to tighten the terminals. Try to pull the wires gently to ensure they are secure. The polarity of the motor may not matter since we can easily change its spinning direction in the Verilog code.

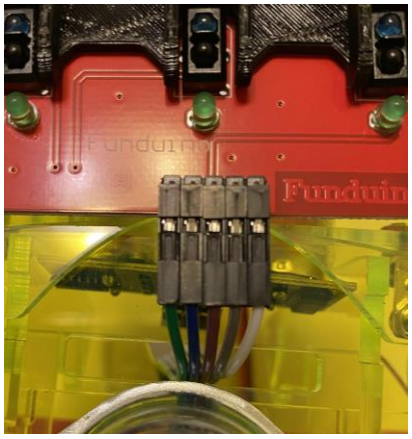


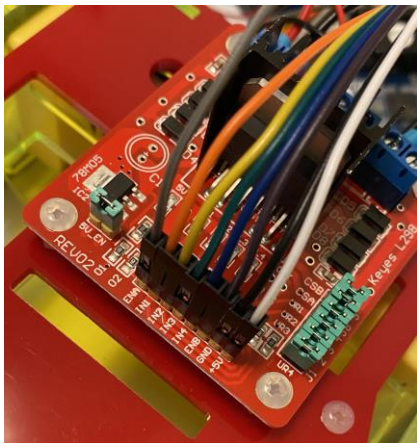
3. Connect the battery wires to the wire terminals in the middle, with **the positive (red) wire connected to VMS and the negative (black) wire connected to GND**. **Caution:** you WILL damage the hardware if you connect them incorrectly.



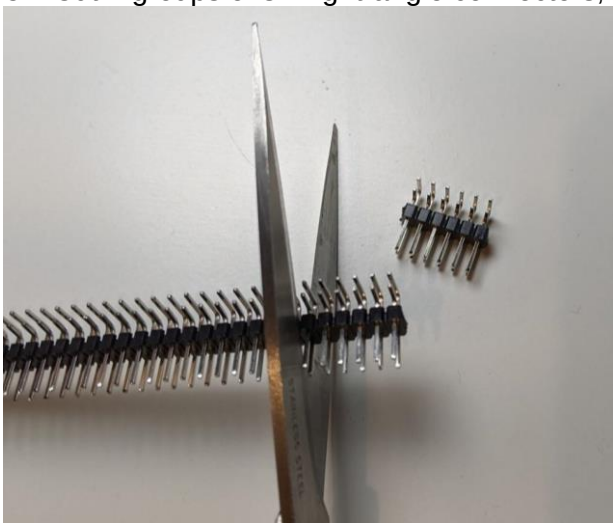
4. Connect jumper wires for the ultrasonic sensor, the 3-way track sensor, and the motor driver.
Note: The jumper wires in each box set may have different colors from the ones in the photos below. Connect them with caution.



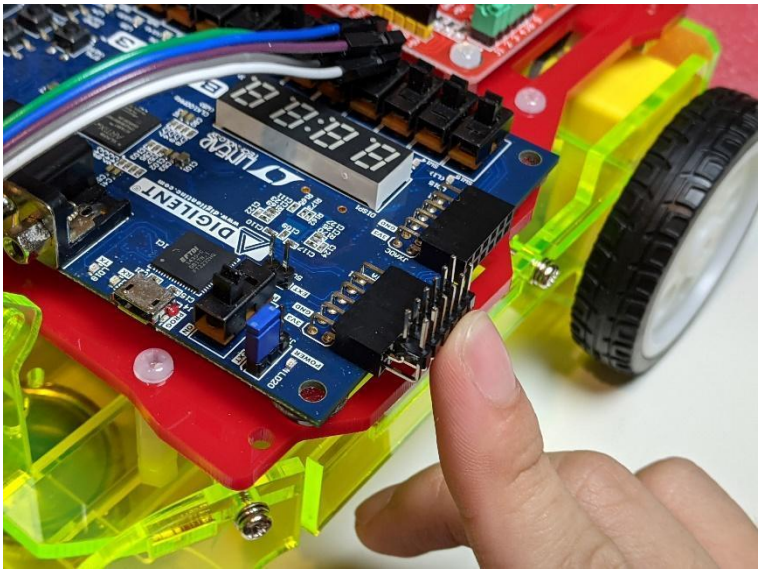




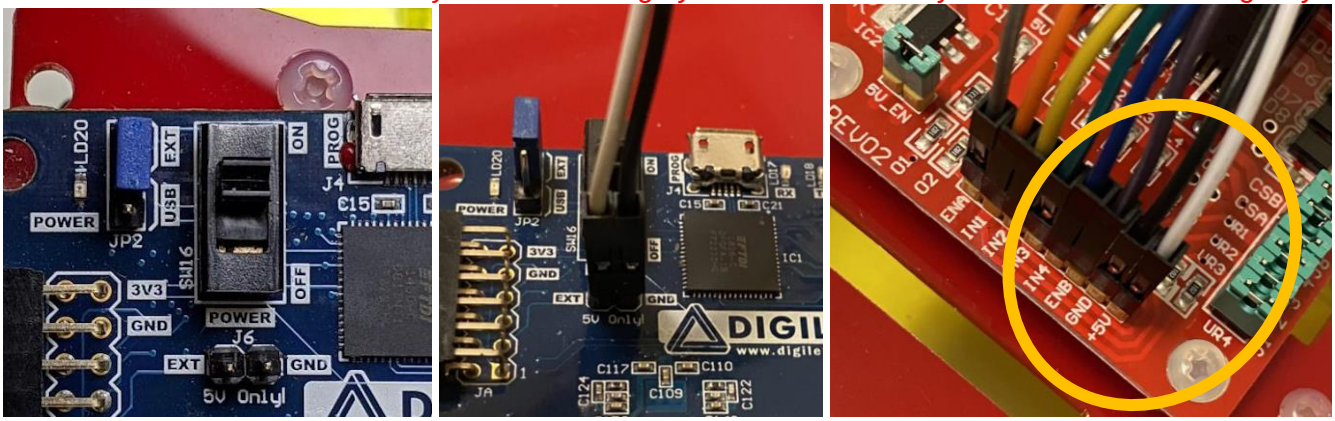
5. Cut 2 groups of six right-angle connectors, if they are not already cut.



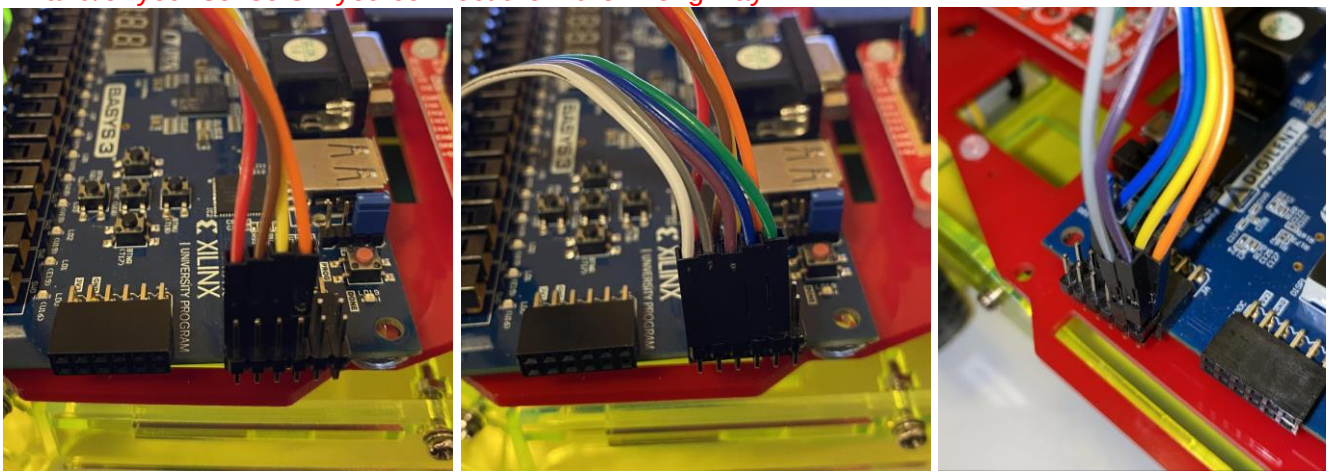
6. Place the FPGA board on top and insert the right-angle pins into the Pmod connectors.



7. Move the power jumper to “EXT” to receive 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, and GND to GND**. **Caution:** you WILL damage your FPGA board if you connect it 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 them the wrong way.



9. Insert the battery, **being careful with the polarity**. **Be sure to double-check all the power-related connections before turning on the power.** Now, flip the power switch on, 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 overcurrent protection state, which prevents the battery from further charging. This may also happen if your duty cycle is too large. 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 so, 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 are allowed to adjust your design and its parameters during the demo.
- Add extra features to assist with debugging and tweaking is encouraged. For example, you can show the distance on the 7-segment display and show the status of the 3-way track sensor on the LEDs.
- Submit one single Verilog file, **lab6_advanced.v**. **Merge the modules from sonic.v, motor.v, and tracker_sensor.v into lab6_advanced.v.** Do not include modules for debounce, one-pulse, or clock divider in the file.
- Do not submit any compressed files, which will be considered an incorrect format.
- Please make sure your kart can operate without being connected to your PC (e.g., via any cords). Consider using USB or SPI Flash programming modes for this purpose.
- DO NOT copy-and-paste code segments from the PDF materials. This can introduce invisible non-ASCII characters, which may lead to difficult-to-debug syntax errors.
- Submit your report in PDF format with a such as **`lab6_advanced_report_group01.pdf`** (where **`01`** is your two-digit group ID).
  - This lab is scored by groups.
  - You must list the group members (with names and student IDs) and describe your job partition and responsibilities clearly in the report.
- Be prepared to answer questions from the TAs during the demo about your design and implementation.
- Use the EECLASS forum to ask questions about the specification.