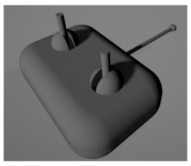
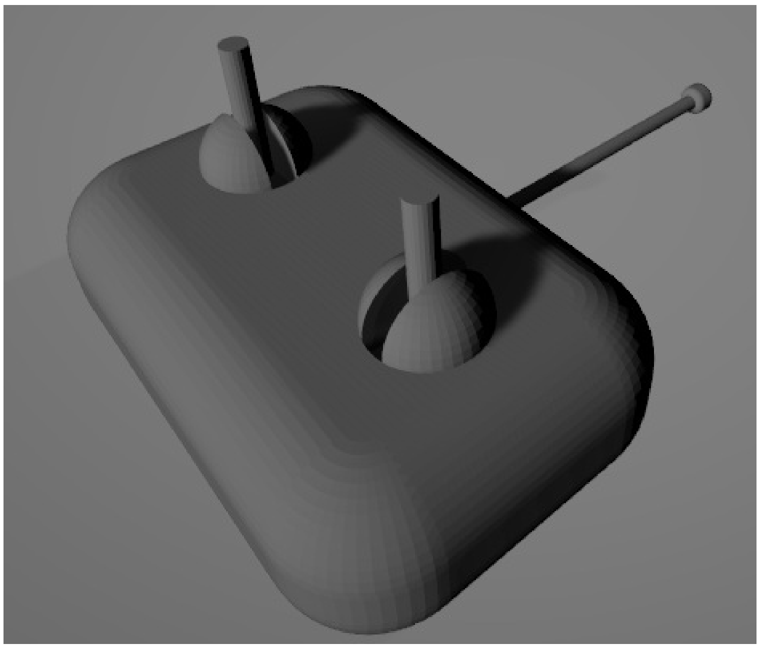


instructables VHDL Motor Speed Control Decide Direction and Speed Left and Right Speed Controller Instructions

Contents

instructables VHDL Motor Speed Control Decide Direction and Speed Left and Right Speed Controller



NOTE: This page is one part of a larger build. Please ensure you start HERE, so you understand where the following fits in within the larger project

Overview

Motor speed and direction control is one of the two main divisions in the photodetector robot, the other one is the photodetector or light detector division. While the photodetector division focuses on the robot's vision, the motor speed and direction control division focuses on the robot's movement. The motor speed and direction control process data given from the photodetector division and gives a physical output in the form of motor movement.

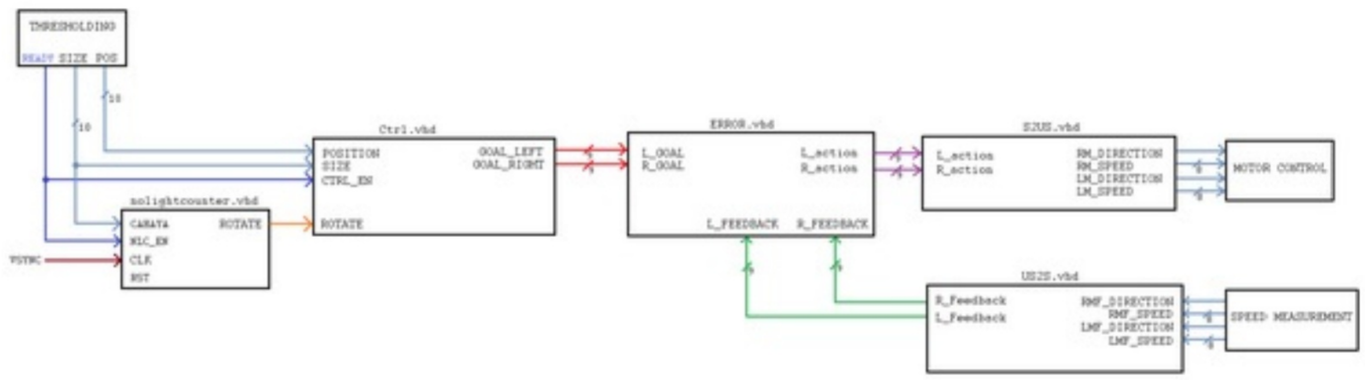
The purpose of this division is to control the speed and direction of both the left and the right motor of the light-seeking robot. To decide these values, you will need the size and the position of the light that had been captured by the camera and processed by thresholding. You will also need the measured speed on each of the motors. From these inputs, you will be able to output the PWM (Pulse-Width Modulation) value for each of the motors.

To achieve this, you will need to make these VHDL modules (also linked below):

1. The control
2. The error calculation
3. The binary conversion
4. The absence of a light source

You can look at the VHDL code for this division here.

Supplies
We recommend to code with ISE Design Suite 14.7 as it can also be used to test the code in VHDL. However, to upload the code into BASYS 3, you will need to install Vivado (ver. 2015.4 or 2016.4) and write the constraint the with .xdc extension.

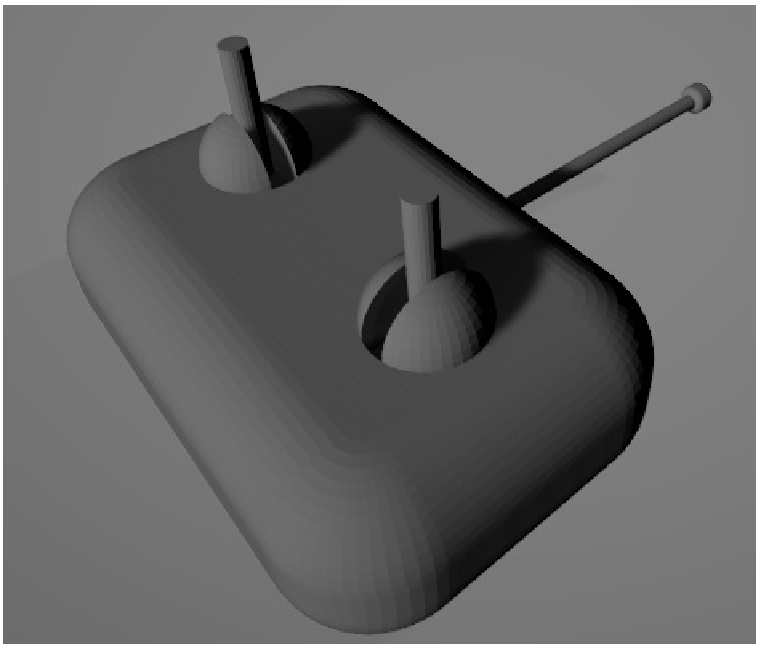


VHDL Motor Speed Control: Decide Direction and Speed, Left and Right Speed Controller: Page 1

INSTRUCTION STEP

Step 1: The Control

To understand how to control the behavior of the light-seeking robot, we will explain the desired behavior of the robot when it sees a light source. This behavior will be controlled according to the position and size of the light source.



The algorithm used is analogous to an RC robot controller, with one lever that can be turned left or right, and another lever that can be turned forward or backward.

To seek light, you want this robot to move in a straight line if the position of the light source is right in front of the robot. To do that, you want the same speed on both the left and the right motors. If the light is located on the left side of the robot, you want the right motor to move faster than the left motor so that the robot can turn to the left towards the light. Conversely, if the light is located on the right side of the robot, you want the left motor to move faster than the right motor so that the robot can turn to the right towards the light. This is analogous to the left lever of an RC controller, where you can control whether you want to move the robot left, right, or straight.

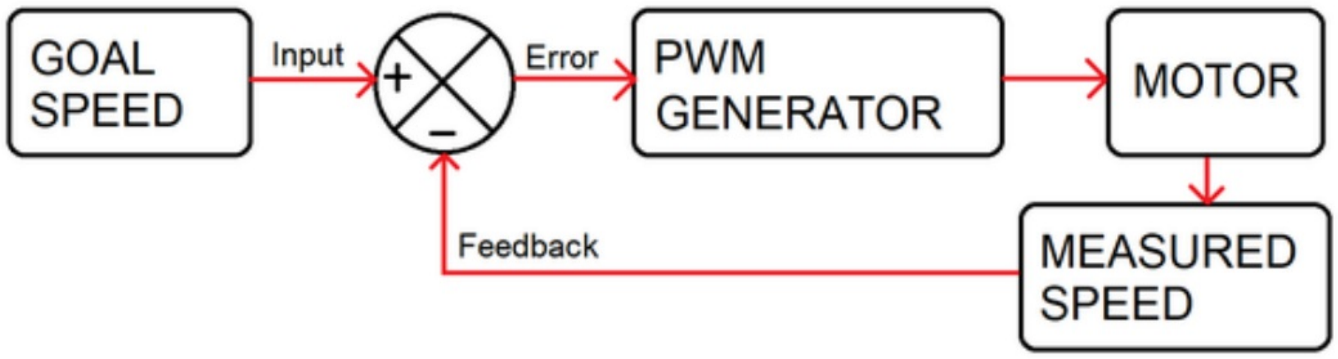
Then, you want the robot to move forward if the light source is far away (small light source), or move backward if the detected light source is too near (big light source). You also want that the farther away the robot is from the light source, the faster the robot moves. This is analogous to the right lever of an RC controller, where you can control whether you want to move forward or backward, and how fast you want it to move.

You can then derive a mathematical formula for the speed of each of the motors, and we choose the speed range between -255 to 255. A negative value means the motor will turn backward, while a positive value means the motor will turn forwards.

That is the basic algorithm for the movement of this robot. To learn more about this module, click here.

Step 2: The Error Calculation
Since you already have the goal speed and direction for the motors, you also want to take into account the measured speed and direction of the motors. If it has reached the speed goal, we want the motor to move solely on its momentum. If it hasn't, we want to add more speed to the motor. In Control theory, this is known as a closed-loop feedback control system.

To learn more about this module, click here.



Step 3: The Binary Conversion
From previous calculations, you've already known the action needed for each of the motors. However, the calculations are done using signed binary. The purpose of this module is to convert these signed values into a value that can be read by the PWM generator, which are the direction (either clockwise or counter-clockwise) and the speed (ranging between 0 to 255). Also, since the feedback from the motor is measured in unsigned binary, another module is needed to convert the unsigned values (direction and speed) into a signed value that can be calculated by the error calculation module. To learn more about this module, click here.

Step 4: The Absence of Light Source
You've made a robot that moves to seek light when light is detected by the robot. But what happens when the robot does not detect light? The purpose of this module is to dictate what to do when such a condition is present.

The easiest way to and a light source to seek is for the robot to rotate in place. After rotating for a set number of seconds, if the robot still hasn't found a light source, you want the robot to stop moving, in order to save power. After another set number of seconds, the robot should rotate in place again to seek the light. To learn more about this module, click here.

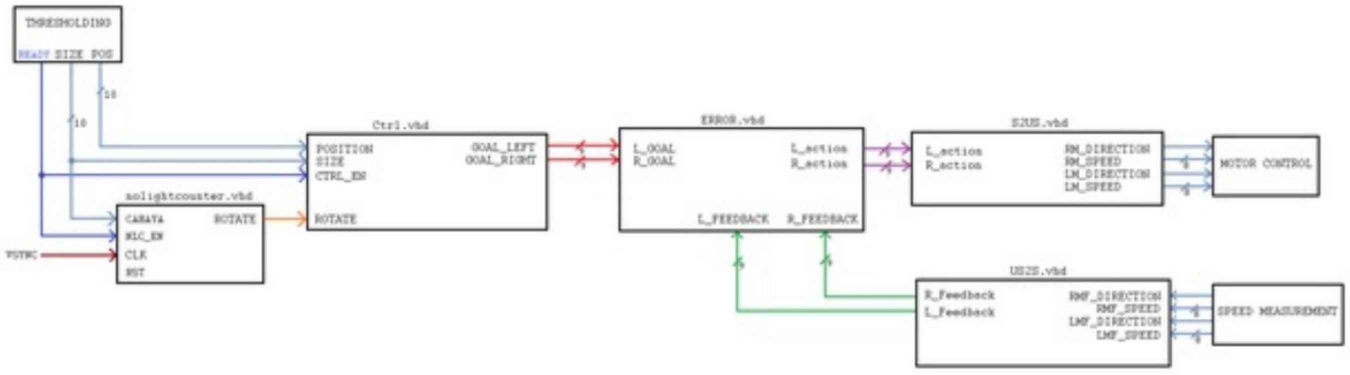
Step 5: How It Works
You can refer to the picture above for this explanation. As mentioned at the start of this instructable, you will need the inputs "size" and "position" from the thresholding division. To make sure that these inputs were valid (for example, when you receive size = 0, size is truly zero because the camera does not detect light, and not because the camera was still initializing) you will also need some kind of indicator, which we call "READY". These data will be processed by the control (Ctrl. vhd) to determine the goal speed of each motor (9 bits, signed).

For a more stable output on the motor, you want to use feedback in a closed-loop system. This requires the inputs"direction" and "speed" of each motor from the motor speed measurement division. Since you want to include these inputs to your calculations, you will have to convert these unsigned values into 9-bit signed binary. This is done by the unsigned to the signed binary converter (US2S.vhd).

What the error calculation (error. vhd) does is subtract the measured speed from the goal speed to determine the action for each motor. This means that when both have the same value, the subtraction becomes zero and the motor will move solely on its momentum. You can also add a factor of multiplication so that the robot might reach the goal speed faster.

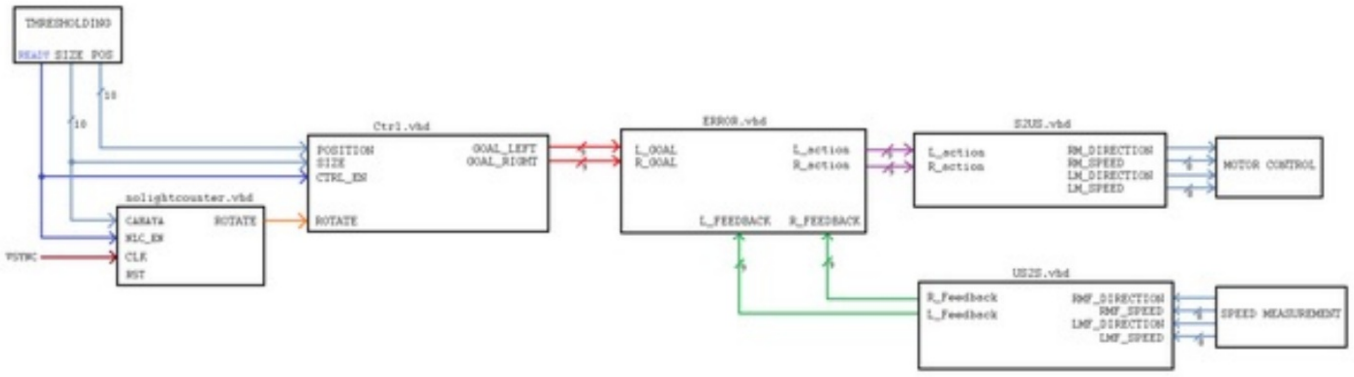
Since the motor controller needs the speed and direction of each motor, you have to translate the signed values of the action into two separate unsigned values: speed (1 bit) and direction (8 bits). This is done by the signed-to-unsigned binary converter (S2US.vhd), and will become inputs to the motor control division.

We also added a module to determine what to do when light isn't detected (no light counter. Bhd). Since this module is basically a counter, it will count how long the robot needs to either rotate or stay in place. This will ensure the robot "sees" its environment rather than just what's in front of it, and conserve battery power when no light source is truly available.



Step 6: Combine the Files

To combine the files, you need to connect the signals from each module. To do that, you have to make a new top level module file. Insert the previous modules' inputs and outputs as components, add signals for the connections and assign each port to the corresponding pair. You can refer to the connections on the illustration above, and look at the code here.



Step 7: Test It

After you've finished with the whole code, you need to know if your code works before you upload it to the board, especially since parts of the code might be made by different people. This requires a testbench, where you will input dummy values and see if the code behaves the way we want it to behave. You can rest start by testing each module, and if they all work correctly, you can then test the top-level module.

Step 8: Try It on the Hardware

After your code has been tested on your computer, you can test the code on the real hardware. You have to make the constraint file on Vivado (.xdc file for BASYS 3) to control which inputs and outputs go to which ports.

IMPORTANT TIP: We learned the hard way that electrical components might have a maximum value of current or voltages. Be sure to refer to the datasheet for the values. For PMOD HB5, be sure to set the voltage from the power source at 12 volts (as this is the required voltage for the motor), and the current as little as needed for the motor to move.

Step 9: Combine It With Other Parts

If the previous steps were successful, combine the code with the other groups for the final code to be uploaded into the robot. Then, voila! You’ve successfully made a light-seeking robot.

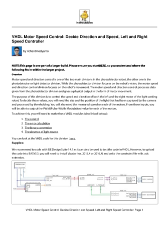
Step 10: Contributors
From left to right:

- Antonius Gregorius Deaven Rivaldi
- Felix Wiguna
- Nicholas Sanjaya
- Richard Medyanto

Very nice: VHDL Motor Speed Control: Decide Direction and Speed, Left and Right Speed Controller: Page 6
Thank you for reviewing! This project is actually only one part of a class project (Light Seeking Robot with BASYS 3 board and OV7670 camera), so I’ll add the link to the class’ instructable soon!

Awesome: I’m looking forward to seeing everything put together.

Documents / Resources

	**instructables VHDL Motor Speed Control Decide Direction and Speed Left and Right Speed Controller** [pdf] Instructions VHDL Motor Speed Control Decide Direction and Speed Left and Right Speed Controller, VHDL Motor Speed, Control Decide Direction and Speed Left and Right Speed Controller

References

- Yours for the making - Instructables
- Richardmedyanto's Profile - Instructables
- VHDL Motor Speed Control: Decide Direction and Speed, Left and Right Speed Controller : 10 Steps - Instructables
- Digital_System123/Ctrl.vhd at main · Skykatzz/Digital_System123 · GitHub
- Digital_System123/error.vhd at main · Skykatzz/Digital_System123 · GitHub
- Digital_System123/nolightcounter.vhd at main · Skykatzz/Digital_System123 · GitHub
- Digital_System123/S2US.vhd at main · Skykatzz/Digital_System123 · GitHub
- Digital_System123/toplevelspeedndir.vhd at main · Skykatzz/Digital_System123 · GitHub
- Digital_System123/US2S.vhd at main · Skykatzz/Digital_System123 · GitHub
- Digital_System123/direction_speed at main · Skykatzz/Digital_System123 · GitHub

- Part 3: Binary Conversion : 6 Steps - Instructables
- Light Seeking Mobile Robot - BASYS 3 (VHDL) : 7 Steps - Instructables
- Part 1: the Control : 16 Steps - Instructables
- VHDL Motor Speed Control : Motor Speed Driver & Measurement : 9 Steps - Instructables
- Part 4: the Absence of Light : 5 Steps - Instructables
- Part 2 : Error Calculation : 3 Steps - Instructables
- VHDL Light Source Detection : Thresholding & Light Source Detection : 11 Steps - Instructables