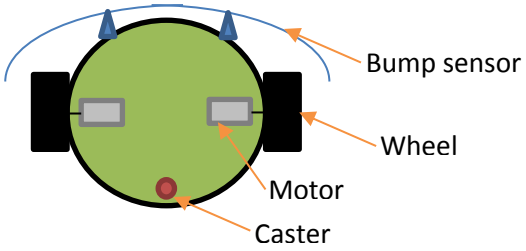
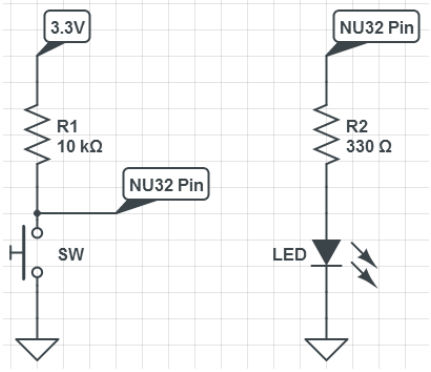
Design Competition 2013 – Milestone 2
Demonstrate to Nick by Wed. 1/30/2013

You goal is to write some code for a robot (that you don't have yet). The robot is differential drive – it has 2 wheels and a caster. The robot has 2 bump sensors, one on the left, one on the right, and overlap in the front.



Hardware
Build two push button circuits, and 4 LED circuits, on the NU32 breadboard.

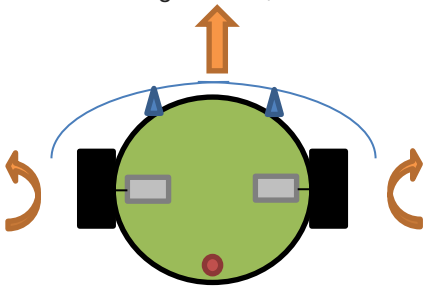


- The push button circuit needs the resistor to guarantee that there is always a voltage for the NU32 to read, otherwise the voltage would drift and be unreliable. In this "pull-up" convention, the NU32 pin is always 3.3V ("high"), and 0V ("low") when the button is pushed. We do it like this to make it easier to tell if the connection is broken, and to stretch your mind a little.
- The LED circuit uses an in-series current-limiting resistor with each LED. The current will be (3.3V - 1.7V / 330Ω) = 5mA. This is for safety – to prevent burning out the LED or the NU32 pin.
- Use E8 and E9 for the push button NU32 pins, and D0, D1, D10, and D11 for the LEDs

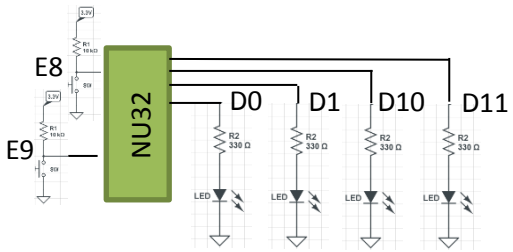
The push buttons represent the bump sensors on the robot. Call the button on E8 the left one, or BL, and the button on E9 the right one, or BR.

The LEDs represent the state of the motors. Call the LED on D0 the left motor, or ML. If ML is on, the motor would be on. If ML is off, the motor would be off. Call the LED on D1 the right motor, or MR.

Call the LED on D10 the direction of the left motor, or DL. If DL is on, the motor would spin to move the robot forwards. If DL is off, the motor would spin to move the robot backwards. But only if ML is on! Call the LED on D11 the direction of the right motor, or DR.

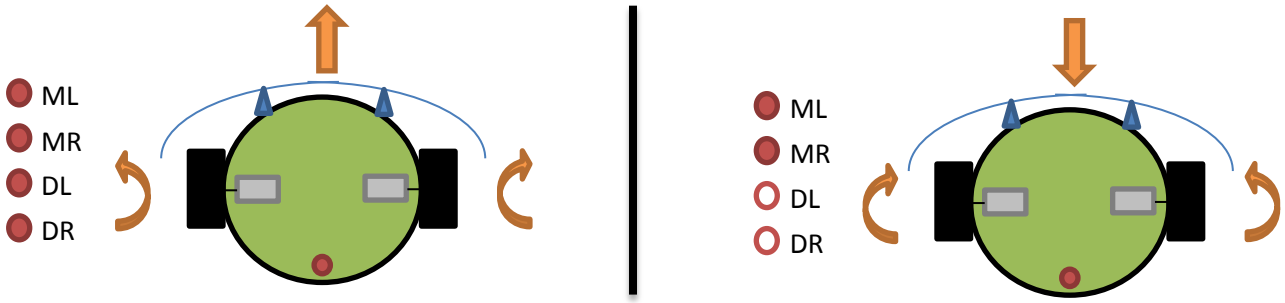


Robot you don't have

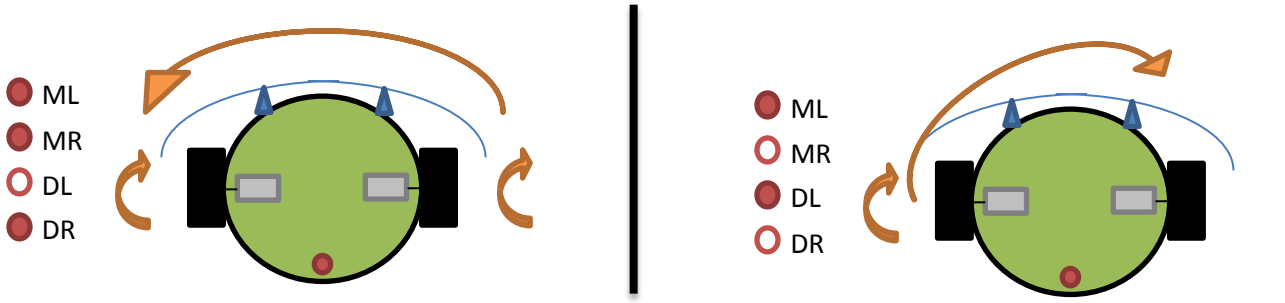


NU32 with push buttons and LEDs

You can go straight by turning on ML, MR, DL, and DR. You can go backwards by turning ML and ML on, and DL and DR off.



You can turn left on a dime by making both ML, MR, and DR on, and DL off. You can make a simple turn to the right by turning on ML and DL, and turning MR and DR off.



Make sense?

Software

Make a project in MPLAB X for the NU32, using DC2013_milestone2.c, NU32.c, NU32.h, and NU32bootloaded.ld.

Turning the LEDs on and off is simple: set the name, like ML, to 1 for on, and = 0 for off

```
Ex: ML = 1; // the ML LED, on D0, is on
    MR = 0; // the MR LED, on D1, is off
```

Reading the buttons is easy too; the variable name is linked to the state, so by using BL in code, you are inserting a 1 or 0 wherever you typed BL, depending on if the button is un-pushed or pushed, respectively.

```
Ex: if(BL) {
        ML = 1;
    }
    else {
        ML = 0;
    }
```

This will turn on the left motor if the left button is not pushed (remember the pin is high if not pushed), and turn the motor off if the button is pushed.

Assignment

Write code so the robot is always driving forward. If the robot hits something on the left, back up, turn right and continue. If the robot hits something on the right, back up, turn left, and continue. If the robot hits something from the front (both buttons are hit), back up, turn 180, and continue.

Demo it!

Note about turning

How do you know how far the robot has turned? You would need some type of position sensor for that, but this robot does not have one, so you'll have to guess. Turn on a motor and wait for a certain amount of time to pass. Assuming the motor goes at constant velocity, the distance travelled is proportional to time. This is called

“dead reckoning”, and it is horrible! Motor speed is proportional to voltage, and voltage decreases as a battery dies, so using dead reckoning is unreliable.

But it is OK here, since you don't have the robot anyway.

Code hints

Use the core timer functions to keep track of time.

```
Ex: WriteCoreTimer(0); // set the timer to 0
int time = ReadCoreTimer(); // get the timer value -> 40000000 (40M) is 1 second
```

Use && (and), || (or), == (equals) to compare two values. Use &, |, = to set values

Use while() loops with the timer

```
Ex. WriteCoreTimer(0); // set the timer to 0
// turn the left motor on for 1 second
while(ReadCoreTimer() < 40000000) {
        ML = 1;
}
ML = 0; // turn the left motor off
```