The CMPE 118 “Cockroach” Robot

Dept. of Computer Engineering, UCSC

Background:

The CMPE-118 “Cockroach” robot is designed to be an accessible mobile platform to teach you basic state machine programming. This document will serve as an introduction to the functionality of the CMPE-118 “Cockroach.” It includes descriptions of how to drive the ‘roach, use its light sensor to detect changes in ambient light levels, and read its bump sensors.

This document describes the “new” roach robot, however the functionality between the new and old robots are the same, except for the LED array, which is not available on the old roaches.

The CMPE-118 “Cockroach” is a simple two-wheeled direct drive ground mobile robot. The chassis is made of laser cut acrylic, with two roller blades wheels attached to the drive motors at the center of the robot. Skids front and back allow the robot to balance on the two main wheels (but will get the robot stuck if the surface is too rough). The microcontroller is a PIC32 packaged in the same Digilent Uno32 form as used elsewhere in the class.

Basic Roach Layout:

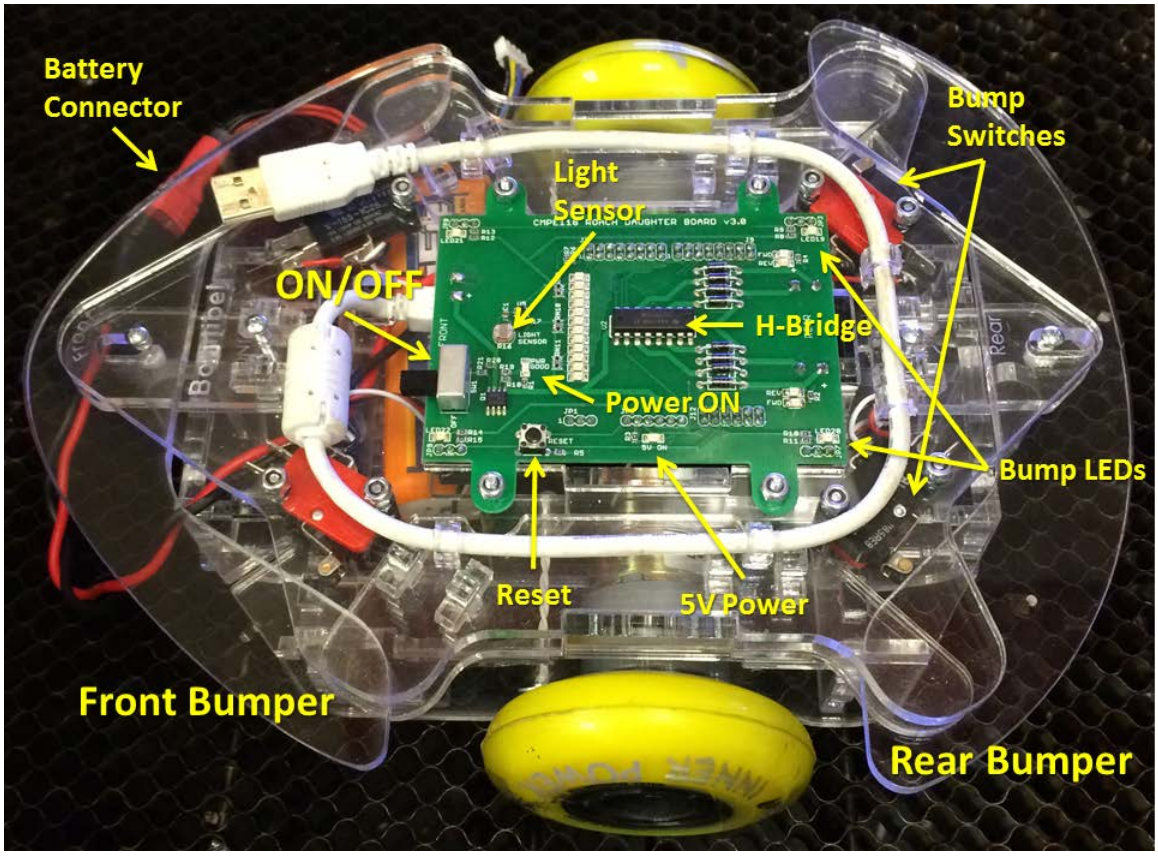


Figure 1: The CMPE-118 "Cockroach" Robot, annotated to show the relevant parts of the chassis.

The basic roach robot is a laser cut acrylic chassis with drive motors and electronics pictured in Figure 1. The front and rear bumpers are floating acrylic pieces, constrained by tabs and a screw/nut combination, that push on snap action lever roller switches. The switches appear as the red/black rectangles with aluminum tabs coming out, and are below the top plate. The roller lever pushes back on the switch plates, restoring them to “unbumped” when pressure is removed from the plate. Indicator LEDs are located at each corner of the electronics board to indicate when the switch is down (not present on old roach design). Note that these are hardware LEDs, and indicate that the switch circuit is working.

There is a main ON/OFF slider switch on the front of the electronics board (on old roach, these are located to the side of the electronics board). It is labeled on the board itself for ON and OFF. If a battery is connected and the switch is turned ON, then POWER GOOD will light indicating that the roach is receiving battery power.

The 5V LED will illuminate when the Uno32 is attached and functional, powered either through the battery (with the switch in the ON position) or when the USB cord is plugged in for programming and debugging. Next to the 5V LED light is the RESET button, which is used to reset the PIC32 processor. Note: battery power is ONLY required to drive the motors. Bump sensors and light sensors will work with just USB power when testing.

The light sensor is an Advanced Photonix Inc PDV-P8001 cadmium sulfide (CdS) photo-resistor, whose resistance changes proportionally to the light falling on its surface. Lower resistance indicates more light, and higher resistance indicates less light. It is passed through a simple OpAmp buffer to ensure no loading of the circuit distorts the sensor, and then directly into the analog to digital converter (ADC) stage of the PIC32.

The H-bridge is a TI SN754410E 1A dual h-bridge that is capable of driving each of the wheels in either direction under software control. There are 8 total flyback diodes to snub the inductive kick from driving the motors, and each motor has a FWD/REV indicator LEDs to visually confirm the direction of rotation. Note that due to the internal manufacturing of the H-bridge, both LEDs will light unless at 100% PWM.

The electronics board has a battery voltage monitor (a 9:1 divider fed directly into an ADC pin) which will monitor the LiFePo4 battery for undervoltage. Software on board will shut down functionality of the roach and echo "Low Battery Voltage!! Recharge your battery" on the serial port. Leaving the roach ON and unattended overnight will kill the battery (~$10/per). Should you need to change the battery, slide it out of the compartment, and pull the Deans connector apart; they are stiff and require a good grip. Get a fully charged battery and plug it in, and slide the battery back into the compartment.

Lastly, the roach has 12 LEDs together in a single line (light-bar) as a user-addressable debugging tool. Each LED can be addressed individually, and functions are provided for access in the Roach software library (detailed below). The old roaches do not have the LED bar, however calling the functions will not result in any ill effects (they will return error if you are checking the return values, see below).

Using the CMPE118 "Cockroach" Robot:

Driving Forwards: Forward motion is implemented by setting both of the Cockroach's motors to the same speed. The left and right motors are controlled by the `Roach_LeftMtrSpeed` and `Roach_RightMtrSpeed` functions, respectively. The functions require integer arguments between 1 and 100 to move the roach forward (note that the roach might not start from a stop at low numbers). Be sure to check your individual roach to make sure both motors turn in the expected direction when you drive them.

Example:
```
Roach_LeftMtrSpeed(80);
Roach_RightMtrSpeed(80);
```

Driving Backwards: Reverse motion is also accomplished by setting the Cockroach motors to the same speed. However, in this case the arguments to `Roach_LeftMtrSpeed` and `Roach_RightMtrSpeed` must be negative, between -1 and -100. The same caveats about motion and polarity apply.

Example:
```
Roach_LeftMtrSpeed(-70);
Roach_RightMtrSpeed(-70);
```

Stopping: Stopping is accomplished by setting both of the Cockroach's motors speeds to 0. This is done with the `Roach_LeftMtrSpeed` and `Roach_RightMtrSpeed` functions.

Example:
```
Roach_LeftMtrSpeed(0);
Roach_RightMtrSpeed(0);
```

Turning: Turning the Cockroach is accomplished by setting different speeds on each of the Cockroach's motors. Turns can be made moving forwards or backwards, depending on the sign of the parameter passed to the motor control functions. Depending on the desired effect, there are various types of turns that can be implemented on the Cockroach (often you will need several of them):

1. Gradual turn: Both motor speeds positive, turns in direction of wheel with slower speed.
2. Pivot turn: One motor speed is set to 0, the other positive. Roach pivots about stopped wheel.
3. Hard turn: One motor positive, other is negative. Roach will turn hard in direction of negative speed wheel.
4. Tank turn: One motor is set to positive number; the other is set to same value but negative. Roach will spin in place in direction of negative number.

Example:
```
Roach_LeftMtrSpeed(80);
Roach_RightMtrSpeed(-80); // tank turn to right (Clockwise)
```

Reading Changes in Battery Voltage: The battery voltage level is read using the `Roach_BatteryVoltage` function, which takes no parameters returns an unsigned integer. The divider results in a 10:1 divide of the value, such that each count of the result is 32mV (e.g.: 272 indicates 8.75V on the battery, at which point the battery should be recharged to prevent damage).

Example:
```
uint16_t currentBat;
currentBat = Roach_BatteryVoltage();
```

Reading Changes in Light Level: The amount of light hitting the Cockroach is obtained by using the `Roach_LightLevel` function, which takes no parameters returns an unsigned integer. The function returns a 10-bit value corresponding to the amount of light seen by the Cockroach's light sensor (with more light being closer to 0 and less light being closer to 1023). A transition from light to darkness, or vice-versa, is sensed by detecting a change in this measured value.

Often, successive values returned by the `Roach_LightLevel` function will vary by quite a few bits due to noise on the sensor. This causes a problem with implementations that rely on discrete measurements, as in the case of detecting light-to-dark (or dark-to-light) transitions. In order to avoid repeated sensing of a transition due to fluctuating values returned by the `Roach_LightLevel` function, it is beneficial to add a tolerance band (hysterisis) to the transition condition.

Hysterisis may be implemented by running two different threshold tests that depend on two values, LIGHT_THRESHOLD and DARK_THRESHOLD. LIGHT_THRESHOLD defines the minimum light level required for a valid "light" condition and DARK_THRESHOLD defines the maximum light level for a valid "dark" condition. If LIGHT_THRESHOLD and DARK_THRESHOLD are slightly lower and higher than the nominal transition point, respectively, false transitions will be minimized. In the example below a light level of 500 is the mid-point of the hysteresis band (note that these numbers will change with every roach):

Example: An event function that tests if the roach has entered the dark may be implemented using hysteresis (with two thresholds) as:

```
#define LIGHT_THRESHOLD 470
#define DARK_THRESHOLD 530
static uint16_t lastLight = 0;

unsigned char TestIfDark(void)
{
        uint8_t goneDark = FALSE;
        goneDark = if((Roach_LightLevel() > DARK_THRESHOLD) &&
                    (lastLight < LIGHT_THRESHOLD));
        if (goneDark) {
```

```
            lastLight = Roach_LightLevel();
        }
        return goneDark;
}
```

The roach's entry into light would then be detected by a TestIfLight function, with the thresholds reversed, thus completing the hysteresis loop. Several other implementations of the hysteresis loop are possible, this is but one (simple) example.

Reading Changes in Bumper State: The state of the Cockroach's bumpers are accessed individually through the functions `Roach_ReadFrontLeftBumper`, `Roach_ReadFrontRightBumper`, `Roach_ReadRearLeftBumper`, and `Roach_ReadRearRightBumper`. These functions return either `BUMPER_TRIPPED` or `BUMPER_NOT_TRIPPED` when called. No debouncing is implemented in these functions, they simply read the appropriate I/O pin and return the value (note that old roaches use Hall effect sensors that usually do not require debouncing, but new ones use mechanical switches which do).

Example:
```
if (Roach_ReadFrontLeftBumper() == BUMPER_TRIPPED) {
 // bumper hit, do something
}
```

In addition to the individual functions, there is a `Roach_ReadBumpers` function which reads all four bump sensors simultaneously and returns a 4-bit value assembled in the following order: front left, front right, rear left, rear right. A 1 in the bit position indicated the switch is `BUMPER_TRIPPED` and a 0 indicates that the corresponding switch is `BUMPER_NOT_TRIPPED`. This function can be extremely useful when implementing switch debouncing on the bumpers.

Example:
```
if (Roach_ReadBumpers() & 0x01) { // use bitwise AND
 // front left bumper hit, do something
}
```

Using the LED light bar: The CMPE-118 "Cockroach" robot includes a diagnostic LED light bar under user control (not available on old roaches). Functions are provided for writing to (and reading from) the LEDs either individually or as a bar graph to show an analog value (in 12 discrete steps). The functions `Roach_LEDSSet` and `Roach_LEDSGet` provide access to each individual LED. `Roach_LEDSSet` takes as a parameter a 12 bit pattern, where a 0 in the pattern causes the corresponding LED to be OFF, and 1 in the pattern causes the corresponding LED to be ON. `Roach_LEDSGet` returns a 12 bit pattern corresponding to a 1 in each bit where the LED is ON, and a 0 corresponding to the LED being OFF (the orthogonal function to `Roach_LEDSSet`).

Example:
```
Roach_LEDSSet(0xAAA); // lights every other LED
Roach_LEDSSet(Roach_LEDSGet()^0x01); // toggles LED_0
```

In addition to the pattern functions for accessing the LEDs, a bar graph function is implemented to light successive LEDs to indicate an analog value. The function `Roach_BarGraph` takes an input value from 0 to 12 and lights the corresponding number of lights from the left with 0 being no lights and 12 being all lit. This is useful for displaying analog values appropriately scaled to indicate what is being read (e.g.: battery voltage).

Example:
```
Roach_BarGraph(8); // lights the leftmost 8 LEDs
```

Full details of all of the functions can be found by examining the Roach.h header file in the C:\CMPE118\include directory. Comments for each public function explain its use and functionality.

CMPE118 “Cockroach” Robot Software Library Test Harness:

The CMPE-118 “Cockroach” robot software library has a built-in test harness (as does every library that is given in the class). This is enabled by adding a #define ROACH_TEST to the project which conditionally compiles the test harness in Roach.c.

The test harness is intended to fully test the roach hardware, and demonstrate that all of the hardware is working correctly. Loading the test harness .hex file onto a roach and resetting it (either using the reset button or via the ds30Loader), will begin the test harness execution.

The test harness will first print out the serial port (setting at 115200 8-N-1):

```
Welcome the the CMPE118 Roach Test Harness
This code will allow someone to confirm operational hardware and software of a Roach
```

The LED bar will light sequentially 5 times and then will flash alternating LEDs 5 times to demonstrate functionality of the LED bar graph. It will then print out instructions on the serial port:

```
To test a roach, click a bumper.  Each bumper runs a specific test.
Front Left: display the current battery voltage
Front Right: Display the Light level live
Rear Left: Test left motor
Rear Right: Test right motor
```

Each subsystem test will now be triggered by the corresponding bumper.

The front left bumper (test 1) will flash the LED bar once, and then slave the battery voltage live to the LED bar (with the USB plugged in and the power switch off, the LED bar should remain off. With the power switch on, some of the LEDs should light. In addition, the battery voltage from the ADC will be displayed on the serial port, for example:

```
Battery voltage is 283
Battery Level Test Complete
```

The LED bar will flash once again to indicate that test 1 is complete.

The front right bumper (test 2) will flash the LED bar twice, and slave the light sensor to the LED bar while displaying the ADC value on the serial port. Dark (or covering the light sensor) will drive the LED bar towards fully illuminated, and light will drive the LED bar to off. The light sensor should be reactive, and have the LED bar change to changing illumination on the sensor. In addition the light sensor reading will be printed to the serial port (at a much lower rate):

```
Current Light Level: 589
Current Light Level: 600
Current Light Level: 852
…
Light Level Test Complete
```

The LED bar will flash twice again to indicate that test 2 is complete.

The rear left bumper (test 3) will flash the LED bar three times, and then drive the left motor forward at different speeds (fastest to slowest), the reverse directions and again drive it from fastest to slowest. The LED bar will indicate direction and speed in four steps during the progression. The speed changes will be printed to the serial port during the test:

```
Left Motor at 100
Left Motor at 80
Left Motor at 60
Left Motor at 40
Left Motor at 0
Left Motor at -100
Left Motor at -80
Left Motor at -60
Left Motor at -40
```

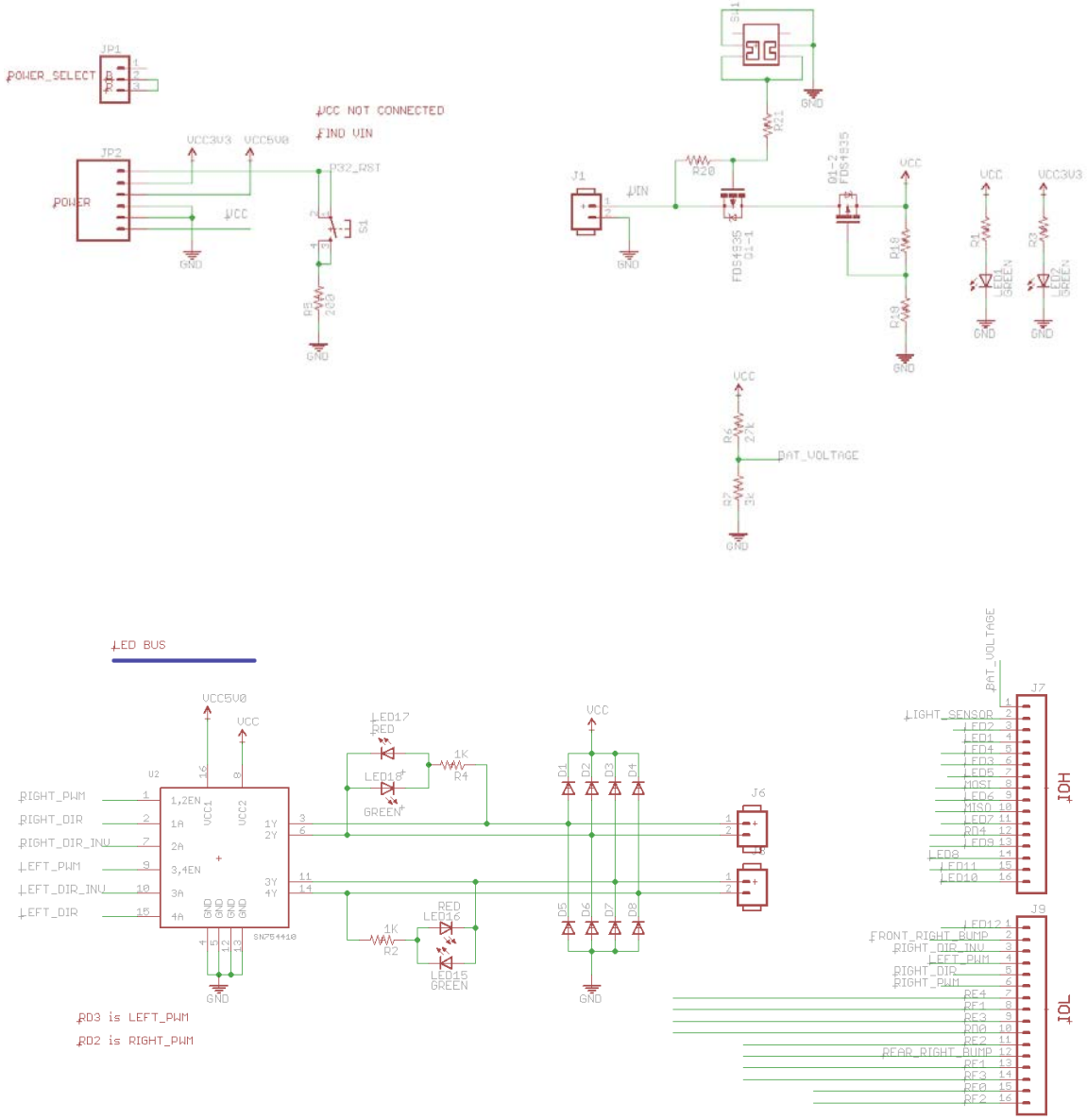
```
Left Motor at 0
Left Motor Test Complete
```

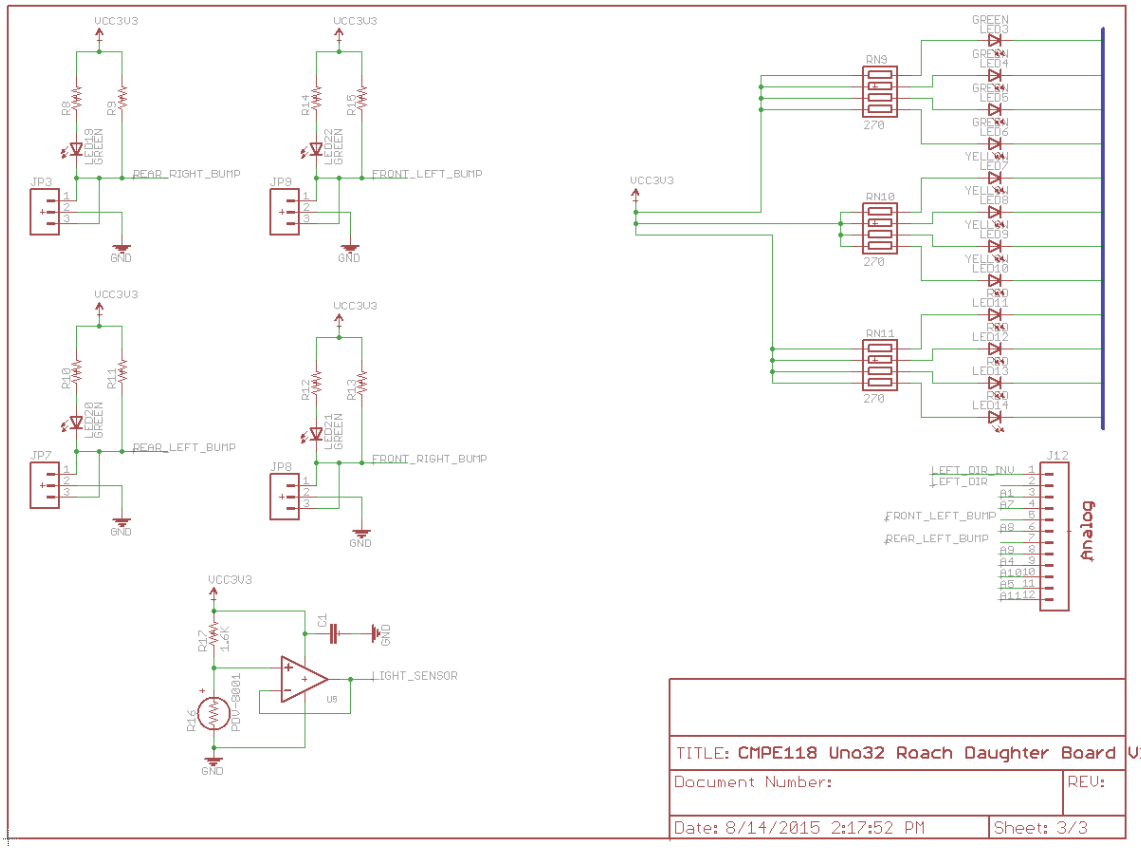
The LED bar will flash three times again to indicate that test 3 is complete.

The rear right bumper (test 4) will flash the LED bar 4 times, and then drive the right motor identically to how the left motor was driven in test 3. When the test is done the LED bar will again flash 4 times to indicate that the test is complete.

With all four tests of the test harness run, the roach is confirmed to have all of its hardware fully functional. If the test is not passed, demonstrate where it fails to a tutor/TA in order to have it repaired. Note that it is a good idea to load the test harness and run it whenever you are working with a new roach to verify that all the hardware is in good working order.

CMPE118 “Cockroach” Robot Schematics:





TITLE: CMPE118 Uno32 Roach Daughter Board V2	
Document Number:	REV:
Date: 8/14/2015 2:17:52 PM	Sheet: 3/3