cs281: Introduction to Computer Systems

Lab04 – K-Maps and a Sensor/Actuator Circuit

Overview

In this lab, we will use the K-maps from last lab to build a circuit for the three lights based on three bit decimal value input, with the system constrained to only operate correctly for input values 000 through 101. Using the Arduino, we will also explore the distance sensor and an actuator, the buzzer. Finally, we will ask you to design and build a solution that combines all three: based on input from the distance sensor, the Arduino control program will control both the buzzer and the three LEDs such that, as we get closer to an object, the buzzer makes a higher frequency pitch and the lights go "up" (more lights are on), and as we get farther from an object, the buzzer makes a lower frequency pitch and the lights go "down" (fewer lights are on).

To help you along, in understanding the devices and the Arduino control, we lead you through exploration of the three pieces individually. The combination, wherein the operation as described above is realized, however, is left up to you.

Your team is required to submit a **Lab Report**. This will be different than your earlier lab writeups, because it will not simply involve answering a sequence of questions. Rather, the Lab Report should be a self-contained and coherent report detailing your design, solution, and implementation of the aggregate objective. Do **not** include a step by step accounting of your exploration steps. The advice and code given to you in the exploration sections is for you to learn about the individual components and the sketch code that can interface with those components. The design, solution, implementation, and report are about how you synthesize and combine the pieces into a whole. Also consider your audience of the report. This should be at a level that, for instance, a Geosciences or Biology professor would understand and appreciate, even without having listened to recent lectures or having read this lab description.

Exploration: LED Ramp Light

As a reminder of the operation of the three LED subsystem: the figure below shows, for each of the decimal input values, the configuration of LEDs. An open circle indicates the LED is off. I chose LED colors of Red, Yellow, and Green in my implementation, but you can implement with other colors if you like. Also note that the LED lighting is "cumulative". So if the decimal input value is 3, for instance, both LED1 (Red) and LED2 (Yellow) are lit. Also note that the same result should occur in *both* cases that the input, interpreted as a decimal value, is either 1 *or* 2, and likewise for inputs 4 *or* 5. Also note that the LED result is not defined for *any other* value of input.

Let's call our inputs A, B, and C, representing the msb to lsb of the input. *LED1* indicates that the first LED should be lit, *LED2* indicates that the second LED should be lit, and *LED3* indicates that the third LED should be lit.

From last lab, you should fill in the K-maps and use what you have learned to circle a minimal set of prime implicants and write down the corresponding boolean expressions for *LED1*, *LED2*, and *LED3*.

Decimal Input	LED Result
0	LED3 LED2 LED1
1, 2	LED3 LED2 LED1
3	LED3 LED2 LED1
4, 5	LED3 LED2 LED1

Figure 1: RampLight

LED1 K-map:

	BC			
A	0 0	0 1	1 1	1 0
0				
1				

LED2 K-map:

	BC			
A	0 0	0 1	1 1	1 0
0				
1				

LED3 K-map:

	BC			
A	0 0	0 1	1 1	1 0
0				
1				

LED Circuit Setup

For the following, we will use a breadboard in conjunction with an Arduino, so you will want to acquire one of the independent "mini" breadboards and wire +5 and GND from the Arduino to the appropriate columns on the breadboard.

We need to have three independent LED circuits, much as we had two LED circuits in our first lab. On the breadboard, you will need three different rows for each of the *LED1*, *LED2*, and *LED3* boolean values will provide input to. Recall from the earlier lab that each input row also has one end of a 330 Ohm resistor. The other end of the resistor is wired in a row with the positive (long lead) side of an LED, and the negative lead of the LED is wired to GND.

The following Arduino code is one way to test your LED setup before proceeding with the RampLight portion. Make sure you understand what it is doing and how it will test your code before blindly entering and running it. Add `println` statements to help your understanding.

```
const int LEDpins[] = {11, 12, 13};

const int WAIT = 500;
int count = 0;
int last = 2;
int index;

void setup(){
  for (int i = 0; i < 3; i++) {
    pinMode(LEDpins[i], OUTPUT);
    digitalWrite(LEDpins[i], LOW);
  }
}

void loop(){
  // take care of making the last LED go Low
  digitalWrite(LEDpins[last], LOW);

  // light the correct LED for this iteration
  index = count % 3;
  digitalWrite(LEDpins[index], HIGH);

  // set up for next iteration
  last = index;
  count++;

  delay(WAIT);
}
```

RampLight Circuit Realization

Based on your boolean equations, and obtaining necessary AND, OR, and NOT chips, implement the three circuits for LED1, LED2, and LED3, wiring the outputs to the rows as defined above. Test and Debug your circuit. You may find the Arduino sketch below helpful. Again, make sure you understand what to expect when you execute the code.

```
const int pinA = 13;
const int pinB = 12;
const int pinC = 11;

const int A[] = {0,0,0,0,1,1};
const int B[] = {0,0,1,1,0,0};
const int C[] = {0,1,0,1,0,1};

const int WAIT = 500;
int count = 0;
int value;

void setup(){
  pinMode(pinA, OUTPUT);
  pinMode(pinB, OUTPUT);
  pinMode(pinC, OUTPUT);
}

void loop(){
  int index;
  index = count++ % 6;  // get a value mod 6, so in range 0 to 5

  digitalWrite(pinA, A[index]);
  digitalWrite(pinB, B[index]);
  digitalWrite(pinC, C[index]);

  delay(WAIT);
}
```

Exploration: Buzzer

In our next exploratory exercise we implement a buzzer, a simple actuator. The buzzer affects physical change in the external environment by introducing a sound which is controlled by the Arduino.

The buzzer has two input pins labeled + and -. There is a metallic membrane inside the buzzer. In its rest state when no voltage is applied across the input pins, the membrane is contracted (bubble in). When a +5 voltage difference is applied across the input pins, the metallic membrane enters an excited state (bubble out). The transition between rest state to excited state or vice versa produces an audible click. By toggling the voltage very quickly, we can cause the buzzer to produce a tone. The frequency of vibration determines the frequency (pitch) of the tone.

Test 1

1. Plug the buzzer onto the breadboard so that the two pins span two distinct rows. Make sure the + pin is in the higher row (up) and the - pin is in the lower row (down).
2. Use a wire to connect the - pin to ground. Plug a longer test wire in the + pin.
3. Now take the other end of the + wire and alternately touch it and release it to any +5v connection. You should year an audible ”click” each time you touch or release the +5v. You can also plug it into the pushbutton on your breadboard to transition back and forth.

Test 2

```
const int buzzerPin = 10;
const int minfreq = 120;
const int freqstep = 100;
const int maxfreq = 1500;
const int timeDelay = 500;

int currentfreq = minfreq;

void setup ()
{
  pinMode(buzzerPin, OUTPUT);
}

void loop()
{
  if (currentfreq > maxfreq) {
    currentfreq = minfreq;
  }
  tone(buzzerPin, currentfreq);

  currentfreq += freqstep;
  delay(timeDelay);
}
```

Test 3

```
const int buzzerPin = 10;
const int delayValues[] = {8,7,6,5,4,3,2,1};
const int stepDelay = 500;

int lastChangeTime;
int index;

void setup ()
{
  pinMode(buzzerPin, OUTPUT);
  lastChangeTime = 0;
  index = 0;
}

void loop ()
{
  if (millis() - lastChangeTime > stepDelay) {
    index = (index + 1) % 8;
    lastChangeTime = millis();
  }

  digitalWrite(buzzerPin, HIGH);
  delay(delayValues[index]);
  digitalWrite(buzzerPin, LOW);
  delay(delayValues[index]);
}
```

Exploration: Ultrasonic Sensor

In this third exploratory section, we learn to use the ultrasonic sensor, aka sonar device. This device measures distance by using sonar; it sends out a ping (ultrasonic – you can't hear it) and records the time the ping echoes back. The time interval, along with the speed of sound, tells you the distance to the object upon which the sound waves echoed.

Obtain a Parallax Ultrasonic Sensor, which has two "microphone" looking things and four pins emitting from the bottom. You will also need a Arduino daughterboard and a battery power cable.

Test 1

1. First set up your daughter board. If it is still in its original packaging, you will need to open the package and mount the breadboard in the center using the adhesive on the back of the breadboard. Just center it somewhere in the middle. It helps to leave the styrofoam on the bottom of the daughterboard so you can press the breadboard on.

2. Now carefully mount the daughterboard onto the top of the Arduino. Be sure the pins line up exactly on both sides before you press. You will fill the rows of pins from the top (away from the Arduino plug) but will notice that a couple of pins remain open near the bottom edge (near the USB plug). Use the labels on common on both boards to make sure of the alignment – you can ruin the device if you get it wrong.

3. Plug your sonar sensor into the daughterboard (see the picture). We want it on the daughterboard instead of the main breadboard because the proximity of other stuff on the breadboard will give you false echo readings. By mounting it on the auxiliary board, you can "aim" it toward an object and be clear of other breadboard structures. Your four pins should span four rows of the breadboard. I chose four pins that were on the Vin and GND side, but away from these connections so that I can access them. See the picture below.

4. On the sonar sensor, connect the + pin (top pin, shown in red above) to +5 (+5 on the daughterboard, NOT the Vin pin) and the - pin (bottom pin, green above) to GND. You will power your Arduino with a battery so you do not need these pins to be wired to the +5V and Ground connections on the main breadboard as we usually do.

5. Connect the Trig pin (yellow) to pin 2 on the Arduino. This will be the pin which turns on the sonic device (an output for your Arduino, an input for your sonar device).

6. Connect the Echo pin (orange) to pin 3 on the Arduino. This will be the pin which reads the echo (input on Arduino, output on sonar device).

7. Connect your 9v battery to the power cable and plug it into the Arduino. Now connect the USB cable from the Arduino to the laptop.

8. Write, upload and run the following program. Open the serial window to see the results. Test the device by aiming the sensor at different objects or by placing objects (a book, a person, etc) in front of it at different distances. How accurate is it? How consistent is it – how stable are the readings? Can you "tune" the distance to be more accurate by playing with a program parameter?

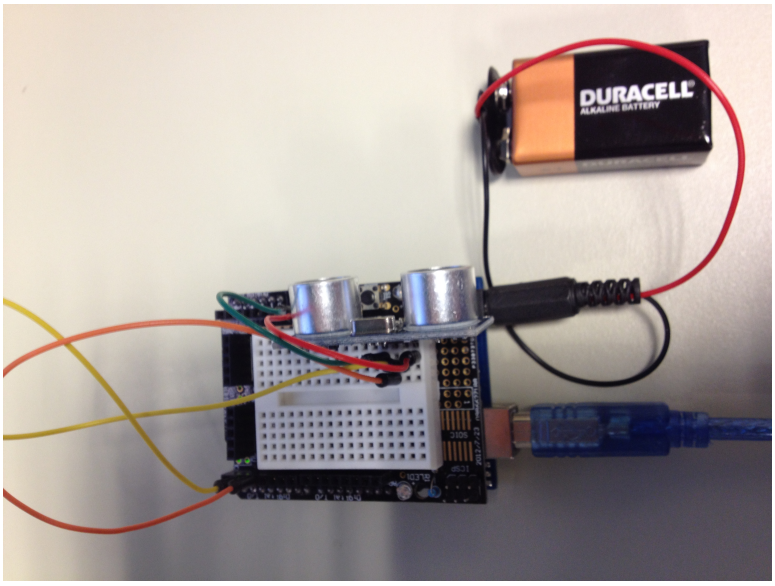

Figure 2: Mounting the Sonar Device

```
#define TRIG 2
#define ECHO 3

void setup ()
{
  pinMode(TRIG,OUTPUT);
  pinMode(ECHO,INPUT);
  Serial.begin(9600);
  Serial.println("Start");
}

void loop ()
{
  Serial.println("Initiating Reading");
  digitalWrite(TRIG,HIGH);
  delay(10);
  digitalWrite(TRIG,LOW);
  int distance = pulseIn(ECHO,HIGH)/2;

  distance = distance / 29;  // tuning parameter - your milage may vary
  Serial.print("Distance in cm is ");
  Serial.println(distance);
  delay(2000);  // wait 2 seconds before next reading
}
```

Design Challenge: Audible Distance Indicator

The goal of this part of the lab is for you to design a solution (hardware and software) to solve a particular design challenge. You are to build a robotic distance detector that emits a sound whose pitch correlates with the distance to a nearby object, and which controls our RampLight so that the light sequence goes ”up” as

the distance to a nearby object goes down.

1. Your device should be sensitive to (respond to) distances that span from very close (1cm or less) to a distance of 300cm (3 meters).
2. Your device should emit a sound tone. Use a low sound tone to indicate a far distance (3m or more). Use a high pitched tone to indicate a near distance (1cm or less). Have your device respond appropriately to intermediate distances.
3. Your device should work with the ramplight circuit to give a visual indication of distance: the closer the distance, the more lights that should be lit, the farther the distance, the fewer lights that should be lit.
4. Make your device fully portable. Yes, you may tether to a computer to download the program, but after that you should be able to operate the device while walking around without tethers to either the laptop or the main breadboard.
5. Do you want a linearly proportional relationship between distance and tone? Are there alternatives to consider?
6. Demonstrate the operation of your device to your professor or lab assistant.
7. Write a lab report in which you fully explain this design challenge and your design solution. Your lab report should stand alone from this lab handout. Fully articulate the design problem and list all your solution steps in your design. Discuss areas where you made design decisions and explain why you opted for the choice you made instead of other alternative designs. Complete one report per group. Report is due the second Friday hence at the start of class as a PDF uploaded to submitbox.

We include a picture of our implementation to show the ability to integrate all of the components in one autonomous system:

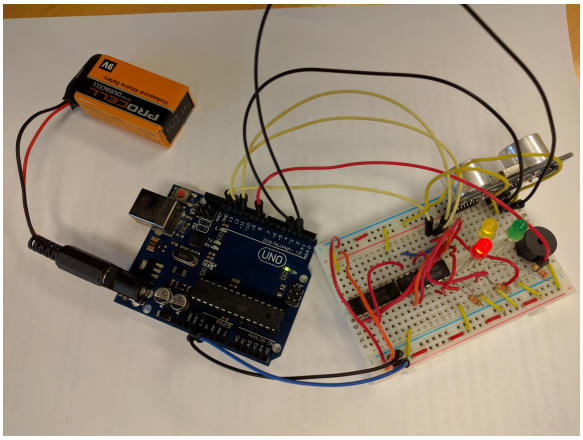

Figure 3: Integrated Sonar Device