# Code Book for Arduino

The code for the scripts has been sourced from a number of resources. It is intended that all scripts for the exercise can be found here. This way teachers can copy and paste the scripts to where they are required, such as in an email to a student.

Where Sparkfun scripts have been used most of the informational details have been removed for convenience. The scripts Sparkfun has done are excellent with the information they provide and looking at the original scripts is a worthwhile exercise.

Contents

[Code Book for Arduino 1](#_Toc501525986)

[Exercise 1 - Blink 3](#_Toc501525987)

[Exercise 2 – Blink 2 4](#_Toc501525988)

[Exercise 3 – Fade 5](#_Toc501525989)

[Exercise 4 – Potentiometer 6](#_Toc501525990)

[Exercise 5 – RGB 7](#_Toc501525991)

[Exercise 6 – Simple traffic Light 10](#_Toc501525992)

[Exercise 7 – PWM 11](#_Toc501525993)

[Exercise 8 – RGB Mood lamp 12](#_Toc501525994)

[Exercise 9 - LED Fire effect 13](#_Toc501525995)

[Exercise 10 – Push Button 14](#_Toc501525996)

[Exercise 11 – Push Buttons 15](#_Toc501525997)

[Exercise 12 – Traffic light with button 16](#_Toc501525998)

[Exercise 13 – Multiple LED’s with chase effect 18](#_Toc501525999)

[Exercise 14 - 21](#_Toc501526000)

[Exercise 15 – LDR Circuit 22](#_Toc501526001)

[Exercise 16 – LDR Flicker 23](#_Toc501526002)

[Exercise 17 – Tone Melody 24](#_Toc501526003)

[Exercise 18 – Tone Multiple 25](#_Toc501526004)

[Exercise 19 – Sparkfun buzzer tune 26](#_Toc501526005)

[Exercise – Darth Vader tune (bonus) 28](#_Toc501526006)

[Exercise 20 – IR Sensor 34](#_Toc501526007)

[Exercise 20A – IF statement & IR Sensor 35](#_Toc501526008)

[Exercise 21 - Flexiforce 36](#_Toc501526009)

[Exercise 21A – Flex Sensor & Servo 37](#_Toc501526010)

[Exercise 22 – Temperature Sensor 38](#_Toc501526011)

[Exercise 23 – Ultrasonic Sensor 39](#_Toc501526012)

[Exercise 24 – Add 2 LED’s, Stop/Go 40](#_Toc501526013)

[Exercise 25 – Servos 41](#_Toc501526014)

[Exercise 26 – Interactive Servos 42](#_Toc501526015)

[Exercise 27 – Motors 43](#_Toc501526016)

[Exercise 28 – Relay 45](#_Toc501526017)

[Exercise 29 – Shift Register 46](#_Toc501526018)

[Exercise 30 – LCD 49](#_Toc501526019)

[Exercise 31 – Fono GSM cellphone build 50](#_Toc501526020)

[Exercise 31 – Simon Says 55](#_Toc501526021)

[Exercise 32, Morse Code transmitter 66](#_Toc501526022)

# Exercise 1 - Blink

/\*

 Blink

 Turns on an LED on for one second, then off for one second, repeatedly.

 Most Arduinos have an on-board LED you can control. On the Uno and

 Leonardo, it is attached to digital pin 13. If you're unsure what

 pin the on-board LED is connected to on your Arduino model, check

 the documentation at http://www.arduino.cc

 This example code is in the public domain.

 modified 8 May 2014

 by Scott Fitzgerald

 \*/

// the setup function runs once when you press reset or power the board

void setup() {

 // initialize digital pin 13 as an output.

 pinMode(13, OUTPUT);

}

// the loop function runs over and over again forever

void loop() {

 digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)

 delay(1000); // wait for a second

 digitalWrite(13, LOW); // turn the LED off by making the voltage LOW

 delay(1000); // wait for a second

}

# Exercise 2 – Blink 2

//Exercise 2, My 1st Project – LED flasher/blink

int ledPin = 9;

Void setup() {

 pinMode (ledPin, OUTPUT);

}

Void loop() {

 digitalWrite(ledPin, HIGH);

 delay(1000);

 digitalWrite(ledPin, LOW);

 delay(1000);

}

# Exercise 3 – Fade

/\*

 Fade

 This example shows how to fade an LED on pin 9

 using the analogWrite() function.

 This example code is in the public domain.

 \*/

int led = 9; // the pin that the LED is attached to

int brightness = 0; // how bright the LED is

int fadeAmount = 5; // how many points to fade the LED by

// the setup routine runs once when you press reset:

void setup() {

 // declare pin 9 to be an output:

 pinMode(led, OUTPUT);

}

// the loop routine runs over and over again forever:

void loop() {

 // set the brightness of pin 9:

 analogWrite(led, brightness);

 // change the brightness for next time through the loop:

 brightness = brightness + fadeAmount;

 // reverse the direction of the fading at the ends of the fade:

 if (brightness == 0 || brightness == 255) {

 fadeAmount = -fadeAmount ;

 }

 // wait for 30 milliseconds to see the dimming effect

 delay(30);

}

# Exercise 4 – Potentiometer

/\*

SparkFun Inventor's Kit

Example sketch 02

POTENTIOMETER

\*/

// Here we're creating a variable called "sensorPin" of type "int"

// and initializing it to have the value "0":

int sensorPin = 0; // The potentiometer is connected to

 // analog pin 0

int ledPin = 13; // The LED is connected to digital pin 13

void setup() // this function runs once when the sketch starts up

{

 // We'll be using pin 13 to light a LED, so we must configure it

 // as an output.

 // Because we already created a variable called ledPin, and

 // set it equal to 13, we can use "ledPin" in place of "13".

 // This makes the sketch easier to follow.

 pinMode(ledPin, OUTPUT);

 // The above line is the same as "pinMode(13, OUTPUT);"

}

void loop() // this function runs repeatedly after setup() finishes

{

 int sensorValue;

 sensorValue = analogRead(sensorPin);

 digitalWrite(ledPin, HIGH); // Turn the LED on

 delay(sensorValue); // Pause for sensorValue

 digitalWrite(ledPin, LOW); // Turn the LED off

 delay(sensorValue); // Pause for sensorValue

}

# Exercise 5 – RGB

/\*

SparkFun Inventor's Kit

Example sketch 03

RGB LED

\*/

const int RED\_PIN = 9;

const int GREEN\_PIN = 10;

const int BLUE\_PIN = 11;

int DISPLAY\_TIME = 100; // In milliseconds

void setup()

{

 pinMode(RED\_PIN, OUTPUT);

 pinMode(GREEN\_PIN, OUTPUT);

 pinMode(BLUE\_PIN, OUTPUT);

}

void loop()

{

 mainColors();

 showSpectrum();

}

void mainColors()

{

 // Off (all LEDs off):

 digitalWrite(RED\_PIN, LOW);

 digitalWrite(GREEN\_PIN, LOW);

 digitalWrite(BLUE\_PIN, LOW);

 delay(1000);

 // Red (turn just the red LED on):

 digitalWrite(RED\_PIN, HIGH);

 digitalWrite(GREEN\_PIN, LOW);

 digitalWrite(BLUE\_PIN, LOW);

 delay(1000);

 // Green (turn just the green LED on):

 digitalWrite(RED\_PIN, LOW);

 digitalWrite(GREEN\_PIN, HIGH);

 digitalWrite(BLUE\_PIN, LOW);

 delay(1000);

 // Blue (turn just the blue LED on):

 digitalWrite(RED\_PIN, LOW);

 digitalWrite(GREEN\_PIN, LOW);

 digitalWrite(BLUE\_PIN, HIGH);

 delay(1000);

 // Yellow (turn red and green on):

 digitalWrite(RED\_PIN, HIGH);

 digitalWrite(GREEN\_PIN, HIGH);

 digitalWrite(BLUE\_PIN, LOW);

 delay(1000);

 // Cyan (turn green and blue on):

 digitalWrite(RED\_PIN, LOW);

 digitalWrite(GREEN\_PIN, HIGH);

 digitalWrite(BLUE\_PIN, HIGH);

 delay(1000);

 // Purple (turn red and blue on):

 digitalWrite(RED\_PIN, HIGH);

 digitalWrite(GREEN\_PIN, LOW);

 digitalWrite(BLUE\_PIN, HIGH);

 delay(1000);

 // White (turn all the LEDs on):

 digitalWrite(RED\_PIN, HIGH);

 digitalWrite(GREEN\_PIN, HIGH);

 digitalWrite(BLUE\_PIN, HIGH);

 delay(1000);

}

void showSpectrum()

{

 int x; // define an integer variable called "x"

 for (x = 0; x < 768; x++)

 // Each time we loop (with a new value of x), do the following:

 {

 showRGB(x); // Call RGBspectrum() with our new x

 delay(10); // Delay for 10 ms (1/100th of a second)

 }

}

void showRGB(int color)

{

 int redIntensity;

 int greenIntensity;

 int blueIntensity;

 if (color <= 255) // zone 1

 {

 redIntensity = 255 - color; // red goes from on to off

 greenIntensity = color; // green goes from off to on

 blueIntensity = 0; // blue is always off

 }

 else if (color <= 511) // zone 2

 {

 redIntensity = 0; // red is always off

 greenIntensity = 255 - (color - 256); // green on to off

 blueIntensity = (color - 256); // blue off to on

 }

 else // color >= 512 // zone 3

 {

 redIntensity = (color - 512); // red off to on

 greenIntensity = 0; // green is always off

 blueIntensity = 255 - (color - 512); // blue on to off

 }

 analogWrite(RED\_PIN, redIntensity);

 analogWrite(BLUE\_PIN, blueIntensity);

 analogWrite(GREEN\_PIN, greenIntensity);

}

# Exercise 6 – Simple traffic Light

//Exercise 6 - Simple Traffic light

// Next lesson will add a button to this, for a pedestrian crossing type of effect

// Sourced from McRoberts, Michael. 'Beginning Arduino. 2nd Ed.'

int ledDelay = 10000; //delay between changes

int redPin = 10;

int yellowPin = 9;

int greenPin = 8;

void setup() {

 pinMode(redPin, OUTPUT);

 pinMode(yellowPin, OUTPUT);

 pinMode(greenPin, OUTPUT);

}

void loop() {

 digitalWrite(redPin, HIGH); //turn the light red on

 delay(ledDelay); //wait 10 seconds as per the ledDelay time

 digitalWrite(yellowPin, HIGH); //turn the yellow on

 delay(2000); //wait 2 seconds

 digitalWrite(greenPin, HIGH); //turn the green on

 digitalWrite(redPin, LOW); //turn the red off

 digitalWrite(yellowPin, LOW); //turn the yellow off

 delay(ledDelay); //delay in milliseconds

 digitalWrite(yellowPin, HIGH); //turn the yellow on

 digitalWrite(greenPin, LOW); //turn the green off

 delay(2000); //delay in milliseconds for 2 seconds

}

# Exercise 7 – PWM

//Exercise 7 using PWM (Pulse width modulation)

// Sourced from McRoberts, Michael. 'Beginning Arduino. 2nd Ed.' project 7 Pulsating lamp

int ledPin =11;

float sinVal;

int ledVal;

void setup() {

 pinMode(ledPin, OUTPUT);

}

void loop() {

 for (int x=0; x<180; x++) {

 //convert degrees to radians then obtain a Sin value

 sinVal = (sin(x\*(3.1412/180))); //note the value for Pi for converting to radians

 ledVal = int (sinVal\*255); //note the 255 for converting to computer binary based numeracy

 analogWrite(ledPin, ledVal);

 delay(25);

 }

}

# Exercise 8 – RGB Mood lamp

//Exercise 8 RGB Mood lamp

// Sourced from McRoberts, Michael. 'Beginning Arduino. 2nd Ed.' Project 8

float RGB1[3];

float RGB2[3];

float INC[3];

int red, green, blue;

int RedPin = 11

int GreenPin = 10

int BluePin = 9

void setup() {

 randomSeed(analogRead(0));

 RGB1[0] = 0;

 RGB1[1] = 0;

 RGB1[2] = 0;

 RGB2[0] = random(256);

 RGB2[1] = random(256);

 RGB2[2] = random(256);

}

void loop() {

 randomSeed(analogRead(0));

 for (int x=0; x<3; x++) {

 INC[x] = (RGB1[x] - RGB2[x])/256; }

 for (int x=0; x<256; x++) {

 red = int(RGB1[0]);

 green = int(RGB1[1]);

 blue = int(RGB1[2]);

 analogWrite (RedPin, red);

 analogWrite (GreenPin, green);

 analogWrite (BluePin, blue);

 delay(100);

 RGB1[0] -= INC[0];

 RGB1[1] -= INC[1];

 RGB1[2] -= INC[2];

 for (int x=0; x<3; x++) {

 RGB2[x] = random(556)-300;

 RGB2[x] = constrain(RGB2[x], 0, 255);

 delay(1000);

 }

}

# Exercise 9 - LED Fire effect

// Exercise 9 - Fire effect

// Sourced from McRoberts, Michael. 'Beginning Arduino. 2nd Ed.' Project 9

int ledPin1 = 9;

int ledPin2 = 10;

int ledPin3 = 11;

void setup()

{

pinMode(ledPin1, OUTPUT);

pinMode(ledPin2, OUTPUT);

pinMode(ledPin3, OUTPUT);

}

void loop()

{

analogWrite(ledPin1, random(120)+135);

analogWrite(ledPin2, random(120)+135);

analogWrite(ledPin3, random(120)+135);

delay(random(100));

}

# Exercise 10 – Push Button

/\*

 Button

 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/Button

 \*/

// constants won't change. They're used here to

// set pin numbers:

const int buttonPin = 2; // the number of the pushbutton pin

const int ledPin = 13; // the number of the LED pin

// variables will change:

int buttonState = 0; // variable for reading the pushbutton status

void setup() {

 // initialize the LED pin as an output:

 pinMode(ledPin, OUTPUT);

 // initialize the pushbutton pin as an input:

 pinMode(buttonPin, INPUT);

}

void loop() {

 // read the state of the pushbutton value:

 buttonState = digitalRead(buttonPin);

 // check if the pushbutton is pressed.

 // if it is, the buttonState is HIGH:

 if (buttonState == HIGH) {

 // turn LED on:

 digitalWrite(ledPin, HIGH);

 }

 else {

 // turn LED off:

 digitalWrite(ledPin, LOW);

 }

}

# Exercise 11 – Push Buttons

/\*

SparkFun Inventor's Kit

Example sketch 05

Version 2.0 6/2012 MDG

\*/

// First we'll set up constants for the pin numbers.

// This will make it easier to follow the code below.

const int button1Pin = 2; // pushbutton 1 pin

const int button2Pin = 3; // pushbutton 2 pin

const int ledPin = 13; // LED pin

void setup()

{

 // Set up the pushbutton pins to be an input:

 pinMode(button1Pin, INPUT);

 pinMode(button2Pin, INPUT);

 // Set up the LED pin to be an output:

 pinMode(ledPin, OUTPUT);

}

void loop()

{

 int button1State, button2State; // variables to hold the pushbutton states

 button1State = digitalRead(button1Pin);

 button2State = digitalRead(button2Pin);

 if (((button1State == LOW) || (button2State == LOW)) // if we're pushing button 1 OR button 2

 && ! // AND we're NOT

 ((button1State == LOW) && (button2State == LOW))) // pushing button 1 AND button 2

 // then...

 {

 digitalWrite(ledPin, HIGH); // turn the LED on

 }

 else

 {

 digitalWrite(ledPin, LOW); // turn the LED off

 }

}

# Exercise 12 – Traffic light with button

//Exercise 12 - Traffic light with button

// Sourced from McRoberts, Michael. 'Beginning Arduino. 2nd Ed.' Project 4

int carRed = 12; // assign the car lights

int carYellow = 11;

int carGreen = 10;

int pedRed = 9; // assign the pedestrian lights

int pedGreen = 8;

int button = 2; // button pin

int crossTime = 5000; // time alloyoud to cross

unsigned long changeTime; // time since button pressed

void setup() {

pinMode(carRed, OUTPUT);

pinMode(carYellow, OUTPUT);

pinMode(carGreen, OUTPUT);

pinMode(pedRed, OUTPUT);

pinMode(pedGreen, OUTPUT);

pinMode(button, INPUT); // button on pin 2

// turn on the green light

digitalWrite(carGreen, HIGH);

digitalWrite(pedRed, HIGH);

}

void loop() {

int state = digitalRead(button);

/\* check if button is pressed and it is over 5 seconds since last button press \*/

if (state == HIGH && (millis() - changeTime) > 5000) {

// Call the function to change the lights

changeLights();

}

}

void changeLights() {

digitalWrite(carGreen, LOW); // green off

digitalWrite(carYellow, HIGH); // yellow on

delay(2000); // wait 2 seconds

digitalWrite(carYellow, LOW); // yellow off

digitalWrite(carRed, HIGH); // red on

delay(1000); // wait 1 second till its safe

digitalWrite(pedRed, LOW); // ped red off

digitalWrite(pedGreen, HIGH); // ped green on

delay(crossTime); // wait for preset time period

// flash the ped green

for (int x=0; x<10; x++) {

digitalWrite(pedGreen, HIGH);

delay(250);

digitalWrite(pedGreen, LOW);

delay(250);

}

// turn ped red on

digitalWrite(pedRed, HIGH);

delay(500);

digitalWrite(carYellow, HIGH); // yellow on

digitalWrite(carRed, LOW); // red off

delay(1000);

digitalWrite(carGreen, HIGH);

digitalWrite(carYellow, LOW); // yellow off

// record the time since last change of lights

changeTime = millis();

// then return to the main program loop

}

# Exercise 13 – Multiple LED’s with chase effect

/\*

SparkFun Inventor's Kit Example sketch 04

MULTIPLE LEDs

Version 2.0 6/2012 MDG

\*/

int ledPins[] = {2,3,4,5,6,7,8,9};

void setup()

{

 int index;

 for(index = 0; index <= 7; index++)

 {

 pinMode(ledPins[index],OUTPUT);

 // ledPins[index] is replaced by the value in the array.

 // For example, ledPins[0] is 2

 }

}

void loop()

{

 oneAfterAnotherNoLoop(); // Light up all the LEDs in turn

 //oneAfterAnotherLoop(); // Same as oneAfterAnotherNoLoop,

 //oneOnAtATime(); // Turn on one LED at a time,

 //pingPong(); // Light the LEDs middle to the edges

 //marquee(); // Chase lights like you see on signs

 //randomLED(); // Blink LEDs randomly

}

void oneAfterAnotherNoLoop()

{

 int delayTime = 100; // time (milliseconds) to pause between LEDs

 // make this smaller for faster switching

 // turn all the LEDs on:

 digitalWrite(ledPins[0], HIGH); //Turns on LED #0 (pin 2)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[1], HIGH); //Turns on LED #1 (pin 3)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[2], HIGH); //Turns on LED #2 (pin 4)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[3], HIGH); //Turns on LED #3 (pin 5)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[4], HIGH); //Turns on LED #4 (pin 6)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[5], HIGH); //Turns on LED #5 (pin 7)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[6], HIGH); //Turns on LED #6 (pin 8)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[7], HIGH); //Turns on LED #7 (pin 9)

 delay(delayTime); //wait delayTime milliseconds

 // turn all the LEDs off:

 digitalWrite(ledPins[7], LOW); //Turn off LED #7 (pin 9)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[6], LOW); //Turn off LED #6 (pin 8)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[5], LOW); //Turn off LED #5 (pin 7)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[4], LOW); //Turn off LED #4 (pin 6)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[3], LOW); //Turn off LED #3 (pin 5)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[2], LOW); //Turn off LED #2 (pin 4)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[1], LOW); //Turn off LED #1 (pin 3)

 delay(delayTime); //wait delayTime milliseconds

 digitalWrite(ledPins[0], LOW); //Turn off LED #0 (pin 2)

 delay(delayTime); //wait delayTime milliseconds

}

void oneAfterAnotherLoop()

{

 int index;

 int delayTime = 100; // milliseconds to pause between LEDs

 for(index = 0; index <= 7; index++)

 {

 digitalWrite(ledPins[index], HIGH);

 delay(delayTime);

 }

 for(index = 7; index >= 0; index--)

 {

 digitalWrite(ledPins[index], LOW);

 delay(delayTime);

 }

}

void oneOnAtATime()

{

 int index;

 int delayTime = 100; // milliseconds to pause between LEDs

 // make this smaller for faster switching

 for(index = 0; index <= 7; index++)

 {

 digitalWrite(ledPins[index], HIGH); // turn LED on

 delay(delayTime); // pause to slow down

 digitalWrite(ledPins[index], LOW); // turn LED off

 }

}

void pingPong()

{

 int index;

 int delayTime = 100; // milliseconds to pause between LEDs

 for(index = 0; index <= 7; index++)

 {

 digitalWrite(ledPins[index], HIGH); // turn LED on

 delay(delayTime); // pause to slow down

 digitalWrite(ledPins[index], LOW); // turn LED off

 }

 for(index = 7; index >= 0; index--)

 {

 digitalWrite(ledPins[index], HIGH); // turn LED on

 delay(delayTime); // pause to slow down

 digitalWrite(ledPins[index], LOW); // turn LED off

 }

}

void marquee()

{

 int index;

 int delayTime = 200; // milliseconds to pause between LEDs

 for(index = 0; index <= 3; index++) // Step from 0 to 3

 {

 digitalWrite(ledPins[index], HIGH); // Turn a LED on

 digitalWrite(ledPins[index+4], HIGH); // Skip four, and turn that LED on

 delay(delayTime); // Pause to slow down the sequence

 digitalWrite(ledPins[index], LOW); // Turn the LED off

 digitalWrite(ledPins[index+4], LOW); // Skip four, and turn that LED off

 }

}

void randomLED()

{

 int index;

 int delayTime;

 index = random(8); // pick a random number between 0 and 7

 delayTime = 100;

 digitalWrite(ledPins[index], HIGH); // turn LED on

 delay(delayTime); // pause to slow down

 digitalWrite(ledPins[index], LOW); // turn LED off

}

# Exercise 14 -

*Code for Project 6*

**byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13}; // Create array for LED pins**

**int ledDelay; // delay between changes**

**int direction = 1;**

**int currentLED = 0;**

**unsigned long changeTime;**

**int potPin = 2; // select the input pin for the potentiometer**

**void setup() {**

**for (int x=0; x<10; x++) { // set all pins to output**

**pinMode(ledPin[x], OUTPUT); }**

**changeTime = millis();**

**}**

**void loop() {**

**ledDelay = analogRead(potPin); // read the value from the pot**

**if ((millis() - changeTime) > ledDelay) { // if it has been ledDelay ms since**

**last change**

**changeLED();**

**changeTime = millis();**

**}**

**}**

**void changeLED() {**

**for (int x=0; x<10; x++) { // turn off all LED's**

**digitalWrite(ledPin[x], LOW);**

**}**

**digitalWrite(ledPin[currentLED], HIGH); // turn on the current LED**

**currentLED += direction; // increment by the direction value**

**// change direction if we reach the end**

**if (currentLED == 9) {direction = -1;}**

**if (currentLED == 0) {direction = 1;}**

**}**

# Exercise 15 – LDR Circuit

/\*

SparkFun Inventor's Kit Example sketch 06

PHOTO RESISTOR

Version 2.0 6/2012 MDG

\*/

const int sensorPin = 0;

const int ledPin = 9;

int lightLevel, high = 0, low = 1023;

void setup()

{

 pinMode(ledPin, OUTPUT);

}

void loop()

{

 lightLevel = analogRead(sensorPin);

 // lightLevel = map(lightLevel, 0, 1023, 0, 255);

 manualTune(); // manually change the range from light to dark

 analogWrite(ledPin, lightLevel);

}

void manualTune()

{

 lightLevel = map(lightLevel, 0, 1023, 0, 255);

 lightLevel = constrain(lightLevel, 0, 255);

}

void autoTune()

{

 if (lightLevel < low)

 {

 low = lightLevel;

 }

 if (lightLevel > high)

 {

 high = lightLevel;

 }

 lightLevel = map(lightLevel, low+30, high-30, 0, 255);

 lightLevel = constrain(lightLevel, 0, 255);

}

# Exercise 16 – LDR Flicker

// LDR - Exercise 16

// blink faster in the dark and slower in the light

// sourced from p49 Karvinen, Kimmo & Karvinen, Tero. 'Make: Getting started with Sensors'

// This sketch was also used for a pot

// to do so change photoPin to potPin

int photoPin=A0;

int ledPin=13;

int x=-1; //0..1023

void setup() {

 pinMode(ledPin, OUTPUT);

}

void loop() {

 x=analogRead(photoPin);

 digitalWrite(ledPin, HIGH);

 delay(x/10);

 digitalWrite(ledPin, LOW);

 delay(x/10);

}

# Exercise 17 – Tone Melody

/\*

 Melody

circuit:

 \* 8-ohm speaker on digital pin 8

created 21 Jan 2010

 modified 30 Aug 2011

 by Tom Igoe

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Tone

\*/

#include "pitches.h"

// notes in the melody:

int melody[] = {

 NOTE\_C4, NOTE\_G3, NOTE\_G3, NOTE\_A3, NOTE\_G3, 0, NOTE\_B3, NOTE\_C4

};

int noteDurations[] = {

 4, 8, 8, 4, 4, 4, 4, 4

};

void setup() {

 // iterate over the notes of the melody:

 for (int thisNote = 0; thisNote < 8; thisNote++) {

 int noteDuration = 1000 / noteDurations[thisNote];

 tone(8, melody[thisNote], noteDuration);

 int pauseBetweenNotes = noteDuration \* 1.30;

 delay(pauseBetweenNotes);

 // stop the tone playing:

 noTone(8);

 }

}

void loop() {

 // no need to repeat the melody.

}

# Exercise 18 – Tone Multiple

/\*

 Multiple tone player

Plays multiple tones on multiple pins in sequence

circuit:

 \* 3 8-ohm speaker on digital pins 6, 7, and 8

created 8 March 2010

 by Tom Igoe

 based on a snippet from Greg Borenstein

This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/Tone4

 \*/

void setup() {

}

void loop() {

 // turn off tone function for pin 8:

 noTone(8);

 // play a note on pin 6 for 200 ms:

 tone(6, 440, 200);

 delay(200);

 // turn off tone function for pin 6:

 noTone(6);

 // play a note on pin 7 for 500 ms:

 tone(7, 494, 500);

 delay(500);

 // turn off tone function for pin 7:

 noTone(7);

 // play a note on pin 8 for 500 ms:

 tone(8, 523, 300);

 delay(300);

}

# Exercise 19 – Sparkfun buzzer tune

/\*

SparkFun Inventor's Kit Example sketch 11

BUZZER

Version 2.0 6/2012 MDG

This sketch uses the buzzer to play songs.

 note frequency

 c 262 Hz

 d 294 Hz

 e 330 Hz

 f 349 Hz

 g 392 Hz

 a 440 Hz

 b 494 Hz

 C 523 Hz

For more information, see http://arduino.cc/en/Tutorial/Tone

\*/

const int buzzerPin = 9;

const int songLength = 18;

char notes[] = "cdfda ag cdfdg gf "; // a space represents a rest

int beats[] = {1,1,1,1,1,1,4,4,2,1,1,1,1,1,1,4,4,2};

int tempo = 150;

void setup()

{

 pinMode(buzzerPin, OUTPUT);

}

void loop()

{

 int i, duration;

 for (i = 0; i < songLength; i++) // step through the song arrays

 {

 duration = beats[i] \* tempo; // length of note/rest in ms

 if (notes[i] == ' ') // is this a rest?

 {

 delay(duration); // then pause for a moment

 }

 else // otherwise, play the note

 {

 tone(buzzerPin, frequency(notes[i]), duration);

 delay(duration); // wait for tone to finish

 }

 delay(tempo/10); // brief pause between notes

 }

 while(true){}

}

int frequency(char note)

{

 int i;

 const int numNotes = 8; // number of notes we're storing

 char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };

 int frequencies[] = {262, 294, 330, 349, 392, 440, 494, 523};

 for (i = 0; i < numNotes; i++) // Step through the notes

 {

 if (names[i] == note) // Is this the one?

 {

 return(frequencies[i]); // Yes! Return the frequency

 }

 }

 return(0); // We looked through everything and didn't find it,

}

# Exercise – Darth Vader tune (bonus)

// NB: ALL NOTES DEFINED WITH STANDARD ENGLISH NAMES, EXCEPT FROM "A"

// THAT IS CALLED WITH THE ITALIAN NAME "LA" BECAUSE A0,A1...ARE THE ANALOG PINS ON ARDUINO.

// (Ab IS CALLED Ab AND NOT LAb)

// Code s mourced from: http://pasted.co/e525c1b2

// from instructables: http://www.instructables.com/id/How-to-easily-play-music-with-buzzer-on-arduino-Th/

#define C0 16.35

#define Db0 17.32

#define D0 18.35

#define Eb0 19.45

#define E0 20.60

#define F0 21.83

#define Gb0 23.12

#define G0 24.50

#define Ab0 25.96

#define LA0 27.50

#define Bb0 29.14

#define B0 30.87

#define C1 32.70

#define Db1 34.65

#define D1 36.71

#define Eb1 38.89

#define E1 41.20

#define F1 43.65

#define Gb1 46.25

#define G1 49.00

#define Ab1 51.91

#define LA1 55.00

#define Bb1 58.27

#define B1 61.74

#define C2 65.41

#define Db2 69.30

#define D2 73.42

#define Eb2 77.78

#define E2 82.41

#define F2 87.31

#define Gb2 92.50

#define G2 98.00

#define Ab2 103.83

#define LA2 110.00

#define Bb2 116.54

#define B2 123.47

#define C3 130.81

#define Db3 138.59

#define D3 146.83

#define Eb3 155.56

#define E3 164.81

#define F3 174.61

#define Gb3 185.00

#define G3 196.00

#define Ab3 207.65

#define LA3 220.00

#define Bb3 233.08

#define B3 246.94

#define C4 261.63

#define Db4 277.18

#define D4 293.66

#define Eb4 311.13

#define E4 329.63

#define F4 349.23

#define Gb4 369.99

#define G4 392.00

#define Ab4 415.30

#define LA4 440.00

#define Bb4 466.16

#define B4 493.88

#define C5 523.25

#define Db5 554.37

#define D5 587.33

#define Eb5 622.25

#define E5 659.26

#define F5 698.46

#define Gb5 739.99

#define G5 783.99

#define Ab5 830.61

#define LA5 880.00

#define Bb5 932.33

#define B5 987.77

#define C6 1046.50

#define Db6 1108.73

#define D6 1174.66

#define Eb6 1244.51

#define E6 1318.51

#define F6 1396.91

#define Gb6 1479.98

#define G6 1567.98

#define Ab6 1661.22

#define LA6 1760.00

#define Bb6 1864.66

#define B6 1975.53

#define C7 2093.00

#define Db7 2217.46

#define D7 2349.32

#define Eb7 2489.02

#define E7 2637.02

#define F7 2793.83

#define Gb7 2959.96

#define G7 3135.96

#define Ab7 3322.44

#define LA7 3520.01

#define Bb7 3729.31

#define B7 3951.07

#define C8 4186.01

#define Db8 4434.92

#define D8 4698.64

#define Eb8 4978.03

// DURATION OF THE NOTES

#define BPM 120 // you can change this value changing all the others

#define H 2\*Q //half 2/4

#define Q 60000/BPM //quarter 1/4

#define E Q/2 //eighth 1/8

#define S Q/4 // sixteenth 1/16

#define W 4\*Q // whole 4/4

void setup() {

pinMode(8, OUTPUT);

pinMode(9, OUTPUT);

digitalWrite(9,LOW);

}

// the loop routine runs over and over again forever:

void loop() {

 //tone(pin, note, duration)

 tone(8,LA3,Q);

 delay(1+Q); //delay duration should always be 1 ms more than the note in order to separate them.

 tone(8,LA3,Q);

 delay(1+Q);

 tone(8,LA3,Q);

 delay(1+Q);

 tone(8,F3,E+S);

 delay(1+E+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,LA3,Q);

 delay(1+Q);

 tone(8,F3,E+S);

 delay(1+E+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,LA3,H);

 delay(1+H);

 tone(8,E4,Q);

 delay(1+Q);

 tone(8,E4,Q);

 delay(1+Q);

 tone(8,E4,Q);

 delay(1+Q);

 tone(8,F4,E+S);

 delay(1+E+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,Ab3,Q);

 delay(1+Q);

 tone(8,F3,E+S);

 delay(1+E+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,LA3,H);

 delay(1+H);

 tone(8,LA4,Q);

 delay(1+Q);

 tone(8,LA3,E+S);

 delay(1+E+S);

 tone(8,LA3,S);

 delay(1+S);

 tone(8,LA4,Q);

 delay(1+Q);

 tone(8,Ab4,E+S);

 delay(1+E+S);

 tone(8,G4,S);

 delay(1+S);

 tone(8,Gb4,S);

 delay(1+S);

 tone(8,E4,S);

 delay(1+S);

 tone(8,F4,E);

 delay(1+E);

 delay(1+E);//PAUSE

 tone(8,Bb3,E);

 delay(1+E);

 tone(8,Eb4,Q);

 delay(1+Q);

 tone(8,D4,E+S);

 delay(1+E+S);

 tone(8,Db4,S);

 delay(1+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,B3,S);

 delay(1+S);

 tone(8,C4,E);

 delay(1+E);

 delay(1+E);//PAUSE QUASI FINE RIGA

 tone(8,F3,E);

 delay(1+E);

 tone(8,Ab3,Q);

 delay(1+Q);

 tone(8,F3,E+S);

 delay(1+E+S);

 tone(8,LA3,S);

 delay(1+S);

 tone(8,C4,Q);

 delay(1+Q);

 tone(8,LA3,E+S);

 delay(1+E+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,E4,H);

 delay(1+H);

 tone(8,LA4,Q);

 delay(1+Q);

 tone(8,LA3,E+S);

 delay(1+E+S);

 tone(8,LA3,S);

 delay(1+S);

 tone(8,LA4,Q);

 delay(1+Q);

 tone(8,Ab4,E+S);

 delay(1+E+S);

 tone(8,G4,S);

 delay(1+S);

 tone(8,Gb4,S);

 delay(1+S);

 tone(8,E4,S);

 delay(1+S);

 tone(8,F4,E);

 delay(1+E);

 delay(1+E);//PAUSE

 tone(8,Bb3,E);

 delay(1+E);

 tone(8,Eb4,Q);

 delay(1+Q);

 tone(8,D4,E+S);

 delay(1+E+S);

 tone(8,Db4,S);

 delay(1+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,B3,S);

 delay(1+S);

 tone(8,C4,E);

 delay(1+E);

 delay(1+E);//PAUSE QUASI FINE RIGA

 tone(8,F3,E);

 delay(1+E);

 tone(8,Ab3,Q);

 delay(1+Q);

 tone(8,F3,E+S);

 delay(1+E+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,LA3,Q);

 delay(1+Q);

 tone(8,F3,E+S);

 delay(1+E+S);

 tone(8,C4,S);

 delay(1+S);

 tone(8,LA3,H);

 delay(1+H);

 delay(2\*H);

}

# Exercise 20 – IR Sensor

/\*

 AnalogReadSerial

 Reads an analog input on pin 0, prints the result to the serial monitor.

 Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.

 \*/

void setup() {

 Serial.begin(9600);

}

// the loop routine runs over and over again forever:

void loop() {

 int sensorValue = analogRead(A0);

 Serial.println(sensorValue);

 delay(1); // delay in between reads for stability

}

# Exercise 20A – IF statement & IR Sensor

/\*

 Conditionals - If statement

This example demonstrates the use of if() statements.

 It reads the state of a potentiometer (an analog input) and turns on an LED

 only if the potentiometer goes above a certain threshold level. It prints the analog value

 regardless of the level.

The circuit:

 \* potentiometer connected to analog pin 0.

 Center pin of the potentiometer goes to the analog pin.

 side pins of the potentiometer go to +5V and ground

 \* LED connected from digital pin 13 to ground

\* Note: On most Arduino boards, there is already an LED on the board

 connected to pin 13, so you don't need any extra components for this example.

created 17 Jan 2009

 modified 9 Apr 2012

 by Tom Igoe

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/IfStatement

 \*/

// These constants won't change:

const int analogPin = A0; // pin that the sensor is attached to

const int ledPin = 13; // pin that the LED is attached to

const int threshold = 400; // an arbitrary threshold level that's in the range of the analog input

void setup() {

 pinMode(ledPin, OUTPUT);

 Serial.begin(9600);

}

void loop() {

 // read the value of the potentiometer:

 int analogValue = analogRead(analogPin);

 // if the analog value is high enough, turn on the LED:

 if (analogValue > threshold) {

 digitalWrite(ledPin, HIGH);

 }

 else {

 digitalWrite(ledPin, LOW);

 }

 // print the analog value:

 Serial.println(analogValue);

 delay(1); // delay in between reads for stability

}

# Exercise 21 - Flexiforce

/\*

Flex Sensor and LEDs created by ScottC on 23rd May 2011 updated on 16/05/2012.

Exercise 21

-----------------------------------------------------\*/

int flexPin = 0;

int flexposition;

void setup() {

 for (int i=4; i<14; i++){

 pinMode(i, OUTPUT); //sets the led pins 4 to 13 to output

 }

}

void loop(){

 //Ensure to turn off ALL LEDs before continuing

 for (int i=4; i<14; i++){

 digitalWrite(i, LOW);

 }

 Serial.begin(9600);

 Serial.println("sensor: ");

 Serial.print(flexposition);

 flexposition = analogRead(flexPin);

int flexReading = map(analogRead(flexPin), 130, 275, 4, 13);

int LEDnum = constrain(flexReading, 4, 13);

blink(LEDnum, 10,1);

}

// The blink function - used to turn the LEDs on and off

void blink(int LEDPin, int onTime, int offTime){

digitalWrite(LEDPin, HIGH);

delay(onTime);

 digitalWrite(LEDPin, LOW);

 delay(offTime);

}

# Exercise 21A – Flex Sensor & Servo

/\*

SparkFun Inventor's Kit Example sketch 09

FLEX SENSOR

Version 2.0 6/2012 MDG

\*/

#include <Servo.h>

Servo servo1;

const int flexpin = 0;

void setup()

{

 Serial.begin(9600);

 servo1.attach(9);

}

void loop()

{

 int flexposition; // Input value from the analog pin.

 int servoposition; // Output value to the servo.

 flexposition = analogRead(flexpin);

 servoposition = map(flexposition, 600, 900, 0, 180);

 servoposition = constrain(servoposition, 0, 180);

 servo1.write(servoposition);

 Serial.print("sensor: ");

 Serial.print(flexposition);

 Serial.print(" servo: ");

 Serial.println(servoposition);

 delay(20); // wait 20ms between servo updates

}

# Exercise 22 – Temperature Sensor

/\*

SparkFun Inventor's Kit Example sketch 07

TEMPERATURE SENSOR

 Use the "serial monitor" window to read a temperature sensor.

 More information on the sensor is available in the datasheet:

http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/TMP35\_36\_37.pdf

Hardware connections:

 5V, SIGNAL, and GND.

 Connect the 5V pin to 5 Volts (5V).

 Connect the SIGNAL pin to ANALOG pin 0.

 Connect the GND pin to ground (GND).

Version 2.0 6/2012 MDG

\*/

const int temperaturePin = 0;

void setup()

{

 Serial.begin(9600);

}

void loop()

{

 float voltage, degreesC, degreesF;

 voltage = getVoltage(temperaturePin);

 degreesC = (voltage - 0.5) \* 100.0;

 degreesF = degreesC \* (9.0/5.0) + 32.0;

 Serial.print("voltage: ");

 Serial.print(voltage);

 Serial.print(" deg C: ");

 Serial.print(degreesC);

 Serial.print(" deg F: ");

 Serial.println(degreesF);

 delay(1000); // repeat once per second (change as you wish!)

}

float getVoltage(int pin)

{

 return (analogRead(pin) \* 0.004882814);

}

# Exercise 23 – Ultrasonic Sensor

// Exercise 23

// Sourced from p58 Kimmo and Tero Karvinen, Make: Getting started with sensors, p58

// HC-SR04 Ultrasonic sensor for distance

int trigPin = 8; //When set to HIGH this will trigger the sensor to emit a pulse

int echoPin = 7; //the echo pin will lsiten for the response to the triggers pulse

float v=343.21+0.6\*20;

// the speed of sound is 343.21 meters per second at 20 degrees C

// the calc also sets the temp as 20 degrees C

// for a slight gain in accuracy - what is the room temperature and what is the speed of sound at that temp?

void setup() {

 Serial.begin(9600);

 pinMode(trigPin,OUTPUT); //set the trigger to pulse

 pinMode(echoPin, INPUT); //set the echo to read the pulse return

}

float distanceCm(){ //define a new function, in 2 parts, one for the trigger (pulse) and one for the echo (return)

 digitalWrite(trigPin, LOW); //set to zero to start

 delayMicroseconds(3); //rapid pulses

 digitalWrite(trigPin, HIGH);

 delayMicroseconds(5);

 digitalWrite(trigPin, LOW);

 float tUs = pulseIn(echoPin, HIGH);

 float t = tUs / 1000.0 / 1000.0 / 2.0; //convert microseconds to seconds

 float d = t\*v; //distance = time x velocity

 return d\*100; //to give a result in cm

}

void loop() {

 int d=distanceCm();

 Serial.println(d, DEC);

 delay (200);

}

# Exercise 24 – Add 2 LED’s, Stop/Go

// Exercise 24

// Using HC-SR04 Ping distance sensor to activate 2 LED's

// LED1 = Red on Pin 11, LED2 = Green on Pin 10

#define trigPin 8

#define echoPin 7

#define led1 11

#define led2 10

void setup() {

 Serial.begin (9600);

 pinMode(trigPin, OUTPUT);

 pinMode(echoPin, INPUT);

 pinMode(led1, OUTPUT);

 pinMode(led2, OUTPUT);

}

void loop() {

 long duration, distance;

 digitalWrite(trigPin, LOW);

 delayMicroseconds(2);

 digitalWrite(trigPin, HIGH);

 delayMicroseconds(10);

 digitalWrite(trigPin, LOW);

 duration = pulseIn(echoPin, HIGH);

 distance = (duration/2) / 29.1;

 if (distance < 10) { // This is where the LED On/Off happens

 digitalWrite(led1,HIGH); // When the Red condition is met, the Green LED should turn off

 digitalWrite(led2,LOW);

}

 else {

 digitalWrite(led1,LOW);

 digitalWrite(led2,HIGH);

 }

 if (distance >= 200 || distance <= 0){

 Serial.println("Out of range");

 }

 else {

 Serial.print(distance);

 Serial.println(" cm");

 }

 delay(500);

}

# Exercise 25 – Servos

/\* Sweep

 by BARRAGAN <http://barraganstudio.com>

 This example code is in the public domain.

 modified 8 Nov 2013

 by Scott Fitzgerald

 http://www.arduino.cc/en/Tutorial/Sweep

\*/

#include <Servo.h>

Servo myservo; // create servo object to control a servo

 // twelve servo objects can be created on most boards

int pos = 0; // variable to store the servo position

void setup()

{

 myservo.attach(9); // attaches the servo on pin 9 to the servo object

}

void loop()

{

 for(pos = 0; pos <= 180; pos += 1) // goes from 0 degrees to 180 degrees

 { // in steps of 1 degree

 myservo.write(pos); // tell servo to go to position in variable 'pos'

 delay(15); // waits 15ms for the servo to reach the position

 }

 for(pos = 180; pos>=0; pos-=1) // goes from 180 degrees to 0 degrees

 {

 myservo.write(pos); // tell servo to go to position in variable 'pos'

 delay(15); // waits 15ms for the servo to reach the position

 }

}

# Exercise 26 – Interactive Servos

/\*

 Controlling a servo position using a potentiometer (variable resistor)

 by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

 modified on 8 Nov 2013

 by Scott Fitzgerald

 http://www.arduino.cc/en/Tutorial/Knob

\*/

#include <Servo.h>

Servo myservo; // create servo object to control a servo

int potpin = 0; // analog pin used to connect the potentiometer

int val; // variable to read the value from the analog pin

void setup()

{

 myservo.attach(9); // attaches the servo on pin 9 to the servo object

}

void loop()

{

 val = analogRead(potpin); // reads the value of the potentiometer (value between 0 and 1023)

 val = map(val, 0, 1023, 0, 180); // scale it to use it with the servo (value between 0 and 180)

 myservo.write(val); // sets the servo position according to the scaled value

 delay(15); // waits for the servo to get there

}

# Exercise 27 – Motors

/\*

SparkFun Inventor's Kit Example sketch 12

SPINNING A MOTOR

Hardware connections:

 Transistor:

 The transistor has three pins. Looking at the flat side with the

 pins down, the order is COLLECTOR, BASE, EMITTER.

 Connect the black wire on the motor to the

 COLLECTOR pin on the transistor.

 Connect the BASE pin through a 330 Ohm resistor to

 digital pin 9.

 Connect the EMITTER pin to GND.

 Motor:

 You've already connected the black wire on the motor to the

 COLLECTOR pin on the transistor.

 Connect the other (red) wire on the motor to 5V.

 Flyback diode:

 When the motor is spinning and suddenly turned off, the

 magnetic field inside it collapses, generating a voltage spike.

 This can damage the transistor. To prevent this, we use a

 "flyback diode", which diverts the voltage spike "around" the

 transistor.

 Connect the side of the diode with the band (cathode) to 5V

 Connect the other side of the diode (anode) to the black wire

 on the motor.

Version 2.0 6/2012 MDG

\*/

const int motorPin = 9;

void setup()

{

 pinMode(motorPin, OUTPUT);

 Serial.begin(9600);

}

void loop()

{

 serialSpeed();

}

void motorOnThenOff()

{

 int onTime = 3000; // milliseconds to turn the motor on

 int offTime = 3000; // milliseconds to turn the motor off

 digitalWrite(motorPin, HIGH); // turn the motor on (full speed)

 delay(onTime); // delay for onTime milliseconds

 digitalWrite(motorPin, LOW); // turn the motor off

 delay(offTime); // delay for offTime milliseconds

}

void motorOnThenOffWithSpeed()

{

 int Speed1 = 200; // between 0 (stopped) and 255 (full speed)

 int Time1 = 3000; // milliseconds for speed 1

 int Speed2 = 50; // between 0 (stopped) and 255 (full speed)

 int Time2 = 3000; // milliseconds to turn the motor off

 analogWrite(motorPin, Speed1); // turns the motor On

 delay(Time1); // delay for onTime milliseconds

 analogWrite(motorPin, Speed2); // turns the motor Off

 delay(Time2); // delay for offTime milliseconds

}

void motorAcceleration()

{

 int speed;

 int delayTime = 20; // milliseconds between each speed step

 for(speed = 0; speed <= 255; speed++)

 {

 analogWrite(motorPin,speed); // set the new speed

 delay(delayTime); // delay between speed steps

 }

 for(speed = 255; speed >= 0; speed--)

 {

 analogWrite(motorPin,speed); // set the new speed

 delay(delayTime); // delay between speed steps

 }

}

void serialSpeed()

{

 int speed;

 Serial.println("Type a speed (0-255) into the box above,");

 Serial.println("then click [send] or press [return]");

 Serial.println(); // Print a blank line

 while(true) // "true" is always true, so this will loop forever.

 {

 while (Serial.available() > 0)

 {

 speed = Serial.parseInt();

 speed = constrain(speed, 0, 255);

 Serial.print("Setting speed to ");

 Serial.println(speed);

 analogWrite(motorPin, speed);

 }

 }

}

# Exercise 28 – Relay

/\*

SparkFun Inventor's Kit Example sketch 13

RELAYS - Use a transistor to drive a relay

Hardware connections:

 Transistor:

 The transistor has three pins. Looking at the flat side with

 the pins down, the order is COLLECTOR, BASE, EMITTER.

 Connect the BASE pin through a 1K resistor to digital pin 2.

 Connect the EMITTER pin to GND.

 Relay coil:

 Connect one side of the coil to the COLLECTOR pin

 on the transistor.

 Connect other side of the coil to 5V.

 Diode:

 Connect the side of the diode with the band (cathode) to 5V

 Connect the other side of the diode (anode) to the COLLECTOR

 pin of the transistor.

Version 2.0 6/2012 MDG

\*/

const int relayPin = 2; // use this pin to drive the transistor

const int timeDelay = 1000; // delay in ms for on and off phases

void setup()

{

 pinMode(relayPin, OUTPUT); // set pin as an output

}

void loop()

{

 digitalWrite(relayPin, HIGH); // turn the relay on

 delay(timeDelay); // wait for one second

 digitalWrite(relayPin, LOW); // turn the relay off

 delay(timeDelay); // wait for one second

}

# Exercise 29 – Shift Register

/\*

SparkFun Inventor's Kit Example sketch 13

SHIFT REGISTER

 The shift register has 16 pins. They are numbered

 counterclockwise starting at the pin 1 mark (notch

 in the end of the chip). See the datasheet above

 for a diagram.

 74HC595 pin LED pin Arduino pin

 1 (QB) LED 2 +

 2 (QC) LED 3 +

 3 (QD) LED 4 +

 4 (QE) LED 5 +

 5 (QF) LED 6 +

 6 (QG) LED 7 +

 7 (QH) LED 8 +

 8 (GND) GND

 9 (QH\*)

 10 (SRCLR\*) 5V

 11 (SRCLK) Digital 3

 12 (RCLK) Digital 4

 13 (OE\*) GND

 14 (SER) Digital 2

 15 (QA) LED 1 +

 16 (VCC) 5V

Version 2.0 6/2012 MDG

\*/

int datapin = 2;

int clockpin = 3;

int latchpin = 4;

byte data = 0;

void setup()

{

 pinMode(datapin, OUTPUT);

 pinMode(clockpin, OUTPUT);

 pinMode(latchpin, OUTPUT);

}

void loop()

{

 oneAfterAnother(); // All on, all off

 //oneOnAtATime(); // Scroll down the line

 //pingPong(); // Like above, but back and forth

 //randomLED(); // Blink random LEDs

 //marquee();

 //binaryCount(); // Bit patterns from 0 to 255

}

void shiftWrite(int desiredPin, boolean desiredState)

{

 bitWrite(data,desiredPin,desiredState);

 shiftOut(datapin, clockpin, MSBFIRST, data);

 digitalWrite(latchpin, HIGH);

 digitalWrite(latchpin, LOW);

}

void oneAfterAnother()

{

 int index;

 int delayTime = 100; // Time (milliseconds) to pause between LEDs

 // Make this smaller for faster switching

 for(index = 0; index <= 7; index++)

 {

 shiftWrite(index, HIGH);

 delay(delayTime);

 }

 for(index = 7; index >= 0; index--)

 {

 shiftWrite(index, LOW);

 delay(delayTime);

 }

}

void oneOnAtATime()

{

 int index;

 int delayTime = 100; // Time (milliseconds) to pause between LEDs

 for(index = 0; index <= 7; index++)

 {

 shiftWrite(index, HIGH); // turn LED on

 delay(delayTime); // pause to slow down the sequence

 shiftWrite(index, LOW); // turn LED off

 }

}

void pingPong()

{

 int index;

 int delayTime = 100; // time (milliseconds) to pause between LEDs

 for(index = 0; index <= 7; index++)

 {

 shiftWrite(index, HIGH); // turn LED on

 delay(delayTime); // pause to slow down the sequence

 shiftWrite(index, LOW); // turn LED off

 }

 for(index = 7; index >= 0; index--)

 {

 shiftWrite(index, HIGH); // turn LED on

 delay(delayTime); // pause to slow down the sequence

 shiftWrite(index, LOW); // turn LED off

 }

}

void randomLED()

{

 int index;

 int delayTime = 100; // time (milliseconds) to pause between LEDs

 index = random(8); // pick a random number between 0 and 7

 shiftWrite(index, HIGH); // turn LED on

 delay(delayTime); // pause to slow down the sequence

 shiftWrite(index, LOW); // turn LED off

}

void marquee()

{

 int index;

 int delayTime = 200; // Time (milliseconds) to pause between LEDs

 for(index = 0; index <= 3; index++)

 {

 shiftWrite(index, HIGH); // Turn a LED on

 shiftWrite(index+4, HIGH); // Skip four, and turn that LED on

 delay(delayTime); // Pause to slow down the sequence

 shiftWrite(index, LOW); // Turn both LEDs off

 shiftWrite(index+4, LOW);

 }

}

void binaryCount()

{

 int delayTime = 1000; // time (milliseconds) to pause between LEDs

 shiftOut(datapin, clockpin, MSBFIRST, data);

 digitalWrite(latchpin, HIGH);

 digitalWrite(latchpin, LOW);

 data++;

 delay(delayTime);

}

# Exercise 30 – LCD

/\*

SparkFun Inventor's Kit Example sketch 15

LIQUID CRYSTAL DISPLAY (LCD)

 1 to GND

 2 to 5V

 3 to the center pin on the potentiometer

 4 to Arduino digital pin 12

 5 to GND

 6 to Arduino digital pin 11

 7 (no connection)

 8 (no connection)

 9 (no connection)

 10 (no connection)

 11 to Arduino digital pin 5

 12 to Arduino digital pin 4

 13 to Arduino digital pin 3

 14 to Arduino digital pin 2

 15 to 5V

 16 to GND

Version 1.0 2/2013 MDG

\*/

#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);

void setup()

{

 lcd.begin(16, 2);

 lcd.clear();

 lcd.print("hello, world!");

}

void loop()

{

 lcd.setCursor(0,1);

 lcd.print(millis()/1000);

}

# Exercise 31 – Fono GSM cellphone build

#include <Adafruit\_ILI9341.h>

#include <Adafruit\_GFX.h>

#include <gfxfont.h>

#include <Adafruit\_FONA.h>

/\*

does:

 \* can make calls on the speaker & mic

\*/

#include <SPI.h>

#include <Wire.h> // this is needed even tho we aren't using it

#include "Adafruit\_GFX.h"

//#include "Adafruit\_ILI9341.h"

#include "Adafruit\_STMPE610.h"

#include <SoftwareSerial.h>

#include "Adafruit\_FONA.h"

#define FONA\_RX 2

#define FONA\_TX 3

#define FONA\_RST 4

SoftwareSerial fonaSS = SoftwareSerial(FONA\_TX, FONA\_RX);

Adafruit\_FONA fona = Adafruit\_FONA(FONA\_RST);

// For the Adafruit TFT shield, these are the default.

#define TFT\_DC 9

#define TFT\_CS 10

// Use hardware SPI (on Uno, #13, #12, #11) and the above for CS/DC

Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC);

// The STMPE610 uses hardware SPI on the shield, and #8

#define STMPE\_CS 8

Adafruit\_STMPE610 ts = Adafruit\_STMPE610(STMPE\_CS);

// This is calibration data for the raw touch data to the screen coordinates

#define TS\_MINX 150

#define TS\_MINY 130

#define TS\_MAXX 3800

#define TS\_MAXY 4000

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* UI details \*/

#define BUTTON\_X 40

#define BUTTON\_Y 100

#define BUTTON\_W 60

#define BUTTON\_H 30

#define BUTTON\_SPACING\_X 20

#define BUTTON\_SPACING\_Y 20

#define BUTTON\_TEXTSIZE 2

// text box where numbers go

#define TEXT\_X 10

#define TEXT\_Y 10

#define TEXT\_W 220

#define TEXT\_H 50

#define TEXT\_TSIZE 3

#define TEXT\_TCOLOR ILI9341\_MAGENTA

// the data (phone #) we store in the textfield

#define TEXT\_LEN 12

char textfield[TEXT\_LEN + 1] = "";

uint8\_t textfield\_i = 0;

// We have a status line for like, is FONA working

#define STATUS\_X 10

#define STATUS\_Y 65

/\* create 15 buttons, in classic candybar phone style \*/

char buttonlabels[15][5] = {"Call", "Clr", "End", "1", "2", "3", "4", "5", "6", "7", "8", "9", "\*", "0", "#" };

uint16\_t buttoncolors[15] = {ILI9341\_GREEN, ILI9341\_DARKGREY, ILI9341\_RED,

 ILI9341\_BLUE, ILI9341\_BLUE, ILI9341\_BLUE,

 ILI9341\_BLUE, ILI9341\_BLUE, ILI9341\_BLUE,

 ILI9341\_BLUE, ILI9341\_BLUE, ILI9341\_BLUE,

 0x061F, ILI9341\_BLUE, 0x061F

 };

Adafruit\_GFX\_Button buttons[15];

// Print something in the mini status bar with either flashstring

void status(const \_\_FlashStringHelper \*msg) {

 tft.fillRect(STATUS\_X, STATUS\_Y, 240, 8, ILI9341\_BLACK);

 tft.setCursor(STATUS\_X, STATUS\_Y);

 tft.setTextColor(ILI9341\_WHITE);

 tft.setTextSize(1);

 tft.print(msg);

}

// or charstring

void status(char \*msg) {

 tft.fillRect(STATUS\_X, STATUS\_Y, 240, 8, ILI9341\_BLACK);

 tft.setCursor(STATUS\_X, STATUS\_Y);

 tft.setTextColor(ILI9341\_WHITE);

 tft.setTextSize(1);

 tft.print(msg);

}

void setup() {

 Serial.begin(9600);

 Serial.println("Arduin-o-Phone!");

 // clear the screen

 tft.begin();

 tft.fillScreen(ILI9341\_BLACK);

 // eep touchscreen not found?

 if (!ts.begin()) {

 Serial.println("Couldn't start touchscreen controller");

 while (1);

 }

 Serial.println("Touchscreen started");

 // create buttons

 for (uint8\_t row = 0; row < 5; row++) {

 for (uint8\_t col = 0; col < 3; col++) {

 buttons[col + row \* 3].initButton(&tft, BUTTON\_X + col \* (BUTTON\_W + BUTTON\_SPACING\_X),

 BUTTON\_Y + row \* (BUTTON\_H + BUTTON\_SPACING\_Y), // x, y, w, h, outline, fill, text

 BUTTON\_W, BUTTON\_H, ILI9341\_WHITE, buttoncolors[col + row \* 3], ILI9341\_WHITE,

 buttonlabels[col + row \* 3], BUTTON\_TEXTSIZE);

 buttons[col + row \* 3].drawButton();

 }

 }

 // create 'text field'

 tft.drawRect(TEXT\_X, TEXT\_Y, TEXT\_W, TEXT\_H, ILI9341\_WHITE);

 status(F("Checking for FONA..."));

 // Check FONA is there

 fonaSS.begin(4800); // if you're using software serial

 // See if the FONA is responding

 if (! fona.begin(fonaSS)) { // can also try fona.begin(Serial1)

 status(F("Couldn't find FONA :("));

 while (1);

 }

 status(F("FONA is OK!"));

 // Check we connect to the network

 while (fona.getNetworkStatus() != 1) {

 status(F("Looking for service..."));

 delay(100);

 }

 status(F("Johns Phone is Now Connected!"));

 // set to external mic & headphone

 fona.setAudio(FONA\_EXTAUDIO);

}

void loop(void) {

 TS\_Point p;

 if (ts.bufferSize()) {

 p = ts.getPoint();

 } else {

 // this is our way of tracking touch 'release'!

 p.x = p.y = p.z = -1;

 }

 // Scale from ~0->4000 to tft.width using the calibration #'s

 if (p.z != -1) {

 p.x = map(p.x, TS\_MINX, TS\_MAXX, 0, tft.width());

 p.y = map(p.y, TS\_MINY, TS\_MAXY, 0, tft.height());

 Serial.print("("); Serial.print(p.x); Serial.print(", ");

 Serial.print(p.y); Serial.print(", ");

 Serial.print(p.z); Serial.println(") ");

 }

 // go thru all the buttons, checking if they were pressed

 for (uint8\_t b = 0; b < 15; b++) {

 if (buttons[b].contains(p.x, p.y)) {

 Serial.print("Pressing: "); Serial.println(b);

 buttons[b].press(true); // tell the button it is pressed

 } else {

 buttons[b].press(false); // tell the button it is NOT pressed

 }

 }

 // now we can ask the buttons if their state has changed

 for (uint8\_t b = 0; b < 15; b++) {

 if (buttons[b].justReleased()) {

 // Serial.print("Released: "); Serial.println(b);

 buttons[b].drawButton(); // draw normal

 }

 if (buttons[b].justPressed()) {

 buttons[b].drawButton(true); // draw invert!

 // if a numberpad button, append the relevant # to the textfield

 if (b >= 3) {

 if (textfield\_i < TEXT\_LEN) {

 textfield[textfield\_i] = buttonlabels[b][0];

 textfield\_i++;

 textfield[textfield\_i] = 0; // zero terminate

 fona.playDTMF(buttonlabels[b][0]);

 }

 }

 // clr button! delete char

 if (b == 1) {

 textfield[textfield\_i] = 0;

 if (textfield > 0) {

 textfield\_i--;

 textfield[textfield\_i] = ' ';

 }

 }

 // update the current text field

 Serial.println(textfield);

 tft.setCursor(TEXT\_X + 2, TEXT\_Y + 5);

 tft.setTextColor(TEXT\_TCOLOR, ILI9341\_BLACK);

 tft.setTextSize(TEXT\_TSIZE);

 tft.print(textfield);

 // its always OK to just hang up

 if (b == 2) {

 status(F("Hanging Up"));

 fona.hangUp();

 delay(200);

 status(F("Call Ended"));

 delay(200);

 status(F("Johns Phone is Now Connected!"));

 }

 // we dont really check that the text field makes sense

 // just try to call

 if (b == 0) {

 status(F("Calling"));

 Serial.print("Calling "); Serial.print(textfield); Serial.println("Phone Volume: " + fona.getVolume());

 fona.callPhone(textfield);

 fona.setAudio(0); //0 for earphones, 1 for device

 }

 delay(220); // UI debouncing

 }

 }

}

# Exercise 31 – Simon Says

/\*

 SparkFun Inventor's Kit Example sketch 16

 Spark Fun Electronics Oct. 7, 2014

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Public Constants

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define NOTE\_B0 31

#define NOTE\_C1 33

#define NOTE\_CS1 35

#define NOTE\_D1 37

#define NOTE\_DS1 39

#define NOTE\_E1 41

#define NOTE\_F1 44

#define NOTE\_FS1 46

#define NOTE\_G1 49

#define NOTE\_GS1 52

#define NOTE\_A1 55

#define NOTE\_AS1 58

#define NOTE\_B1 62

#define NOTE\_C2 65

#define NOTE\_CS2 69

#define NOTE\_D2 73

#define NOTE\_DS2 78

#define NOTE\_E2 82

#define NOTE\_F2 87

#define NOTE\_FS2 93

#define NOTE\_G2 98

#define NOTE\_GS2 104

#define NOTE\_A2 110

#define NOTE\_AS2 117

#define NOTE\_B2 123

#define NOTE\_C3 131

#define NOTE\_CS3 139

#define NOTE\_D3 147

#define NOTE\_DS3 156

#define NOTE\_E3 165

#define NOTE\_F3 175

#define NOTE\_FS3 185

#define NOTE\_G3 196

#define NOTE\_GS3 208

#define NOTE\_A3 220

#define NOTE\_AS3 233

#define NOTE\_B3 247

#define NOTE\_C4 262

#define NOTE\_CS4 277

#define NOTE\_D4 294

#define NOTE\_DS4 311

#define NOTE\_E4 330

#define NOTE\_F4 349

#define NOTE\_FS4 370

#define NOTE\_G4 392

#define NOTE\_GS4 415

#define NOTE\_A4 440

#define NOTE\_AS4 466

#define NOTE\_B4 494

#define NOTE\_C5 523

#define NOTE\_CS5 554

#define NOTE\_D5 587

#define NOTE\_DS5 622

#define NOTE\_E5 659

#define NOTE\_F5 698

#define NOTE\_FS5 740

#define NOTE\_G5 784

#define NOTE\_GS5 831

#define NOTE\_A5 880

#define NOTE\_AS5 932

#define NOTE\_B5 988

#define NOTE\_C6 1047

#define NOTE\_CS6 1109

#define NOTE\_D6 1175

#define NOTE\_DS6 1245

#define NOTE\_E6 1319

#define NOTE\_F6 1397

#define NOTE\_FS6 1480

#define NOTE\_G6 1568

#define NOTE\_GS6 1661

#define NOTE\_A6 1760

#define NOTE\_AS6 1865

#define NOTE\_B6 1976

#define NOTE\_C7 2093

#define NOTE\_CS7 2217

#define NOTE\_D7 2349

#define NOTE\_DS7 2489

#define NOTE\_E7 2637

#define NOTE\_F7 2794

#define NOTE\_FS7 2960

#define NOTE\_G7 3136

#define NOTE\_GS7 3322

#define NOTE\_A7 3520

#define NOTE\_AS7 3729

#define NOTE\_B7 3951

#define NOTE\_C8 4186

#define NOTE\_CS8 4435

#define NOTE\_D8 4699

#define NOTE\_DS8 4978

#define CHOICE\_OFF 0 //Used to control LEDs

#define CHOICE\_NONE 0 //Used to check buttons

#define CHOICE\_RED (1 << 0)

#define CHOICE\_GREEN (1 << 1)

#define CHOICE\_BLUE (1 << 2)

#define CHOICE\_YELLOW (1 << 3)

#define LED\_RED 10

#define LED\_GREEN 3

#define LED\_BLUE 13

#define LED\_YELLOW 5

// Button pin definitions

#define BUTTON\_RED 9

#define BUTTON\_GREEN 2

#define BUTTON\_BLUE 12

#define BUTTON\_YELLOW 6

// Buzzer pin definitions

#define BUZZER1 4

#define BUZZER2 7

// Define game parameters

#define ROUNDS\_TO\_WIN 13 //Number of rounds to succesfully remember before you win. 13 is do-able.

#define ENTRY\_TIME\_LIMIT 3000 //Amount of time to press a button before game times out. 3000ms = 3 sec

#define MODE\_MEMORY 0

#define MODE\_BATTLE 1

#define MODE\_BEEGEES 2

// Game state variables

byte gameMode = MODE\_MEMORY; //By default, let's play the memory game

byte gameBoard[32]; //Contains the combination of buttons as we advance

byte gameRound = 0; //Counts the number of succesful rounds the player has made it through

void setup()

{

 //Setup hardware inputs/outputs. These pins are defined in the hardware\_versions header file

 //Enable pull ups on inputs

 pinMode(BUTTON\_RED, INPUT\_PULLUP);

 pinMode(BUTTON\_GREEN, INPUT\_PULLUP);

 pinMode(BUTTON\_BLUE, INPUT\_PULLUP);

 pinMode(BUTTON\_YELLOW, INPUT\_PULLUP);

 pinMode(LED\_RED, OUTPUT);

 pinMode(LED\_GREEN, OUTPUT);

 pinMode(LED\_BLUE, OUTPUT);

 pinMode(LED\_YELLOW, OUTPUT);

 pinMode(BUZZER1, OUTPUT);

 pinMode(BUZZER2, OUTPUT);

 //Mode checking

 gameMode = MODE\_MEMORY; // By default, we're going to play the memory game

 // Check to see if the lower right button is pressed

 if (checkButton() == CHOICE\_YELLOW) play\_beegees();

 // Check to see if upper right button is pressed

 if (checkButton() == CHOICE\_GREEN)

 {

 gameMode = MODE\_BATTLE; //Put game into battle mode

 //Turn on the upper right (green) LED

 setLEDs(CHOICE\_GREEN);

 toner(CHOICE\_GREEN, 150);

 setLEDs(CHOICE\_RED | CHOICE\_BLUE | CHOICE\_YELLOW); // Turn on the other LEDs until you release button

 while(checkButton() != CHOICE\_NONE) ; // Wait for user to stop pressing button

 //Now do nothing. Battle mode will be serviced in the main routine

 }

 play\_winner(); // After setup is complete, say hello to the world

}

void loop()

{

 attractMode(); // Blink lights while waiting for user to press a button

 // Indicate the start of game play

 setLEDs(CHOICE\_RED | CHOICE\_GREEN | CHOICE\_BLUE | CHOICE\_YELLOW); // Turn all LEDs on

 delay(1000);

 setLEDs(CHOICE\_OFF); // Turn off LEDs

 delay(250);

 if (gameMode == MODE\_MEMORY)

 {

 // Play memory game and handle result

 if (play\_memory() == true)

 play\_winner(); // Player won, play winner tones

 else

 play\_loser(); // Player lost, play loser tones

 }

 if (gameMode == MODE\_BATTLE)

 {

 play\_battle(); // Play game until someone loses

 play\_loser(); // Player lost, play loser tones

 }

}

//The following functions are related to game play only

// Play the regular memory game

// Returns 0 if player loses, or 1 if player wins

boolean play\_memory(void)

{

 randomSeed(millis()); // Seed the random generator with random amount of millis()

 gameRound = 0; // Reset the game to the beginning

 while (gameRound < ROUNDS\_TO\_WIN)

 {

 add\_to\_moves(); // Add a button to the current moves, then play them back

 playMoves(); // Play back the current game board

 // Then require the player to repeat the sequence.

 for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)

 {

 byte choice = wait\_for\_button(); // See what button the user presses

 if (choice == 0) return false; // If wait timed out, player loses

 if (choice != gameBoard[currentMove]) return false; // If the choice is incorect, player loses

 }

 delay(1000); // Player was correct, delay before playing moves

 }

 return true; // Player made it through all the rounds to win!

}

// Play the special 2 player battle mode

// A player begins by pressing a button then handing it to the other player

// That player repeats the button and adds one, then passes back.

// This function returns when someone loses

boolean play\_battle(void)

{

 gameRound = 0; // Reset the game frame back to one frame

 while (1) // Loop until someone fails

 {

 byte newButton = wait\_for\_button(); // Wait for user to input next move

 gameBoard[gameRound++] = newButton; // Add this new button to the game array

 // Then require the player to repeat the sequence.

 for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)

 {

 byte choice = wait\_for\_button();

 if (choice == 0) return false; // If wait timed out, player loses.

 if (choice != gameBoard[currentMove]) return false; // If the choice is incorect, player loses.

 }

 delay(100); // Give the user an extra 100ms to hand the game to the other player

 }

 return true; // We should never get here

}

// Plays the current contents of the game moves

void playMoves(void)

{

 for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)

 {

 toner(gameBoard[currentMove], 150);

 // Wait some amount of time between button playback

 // Shorten this to make game harder

 delay(150); // 150 works well. 75 gets fast.

 }

}

// Adds a new random button to the game sequence, by sampling the timer

void add\_to\_moves(void)

{

 byte newButton = random(0, 4); //min (included), max (exluded)

 // We have to convert this number, 0 to 3, to CHOICEs

 if(newButton == 0) newButton = CHOICE\_RED;

 else if(newButton == 1) newButton = CHOICE\_GREEN;

 else if(newButton == 2) newButton = CHOICE\_BLUE;

 else if(newButton == 3) newButton = CHOICE\_YELLOW;

 gameBoard[gameRound++] = newButton; // Add this new button to the game array

}

//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

//The following functions control the hardware

// Lights a given LEDs

// Pass in a byte that is made up from CHOICE\_RED, CHOICE\_YELLOW, etc

void setLEDs(byte leds)

{

 if ((leds & CHOICE\_RED) != 0)

 digitalWrite(LED\_RED, HIGH);

 else

 digitalWrite(LED\_RED, LOW);

 if ((leds & CHOICE\_GREEN) != 0)

 digitalWrite(LED\_GREEN, HIGH);

 else

 digitalWrite(LED\_GREEN, LOW);

 if ((leds & CHOICE\_BLUE) != 0)

 digitalWrite(LED\_BLUE, HIGH);

 else

 digitalWrite(LED\_BLUE, LOW);

 if ((leds & CHOICE\_YELLOW) != 0)

 digitalWrite(LED\_YELLOW, HIGH);

 else

 digitalWrite(LED\_YELLOW, LOW);

}

// Wait for a button to be pressed.

// Returns one of LED colors (LED\_RED, etc.) if successful, 0 if timed out

byte wait\_for\_button(void)

{

 long startTime = millis(); // Remember the time we started the this loop

 while ( (millis() - startTime) < ENTRY\_TIME\_LIMIT) // Loop until too much time has passed

 {

 byte button = checkButton();

 if (button != CHOICE\_NONE)

 {

 toner(button, 150); // Play the button the user just pressed

 while(checkButton() != CHOICE\_NONE) ; // Now let's wait for user to release button

 delay(10); // This helps with debouncing and accidental double taps

 return button;

 }

 }

 return CHOICE\_NONE; // If we get here, we've timed out!

}

// Returns a '1' bit in the position corresponding to CHOICE\_RED, CHOICE\_GREEN, etc.

byte checkButton(void)

{

 if (digitalRead(BUTTON\_RED) == 0) return(CHOICE\_RED);

 else if (digitalRead(BUTTON\_GREEN) == 0) return(CHOICE\_GREEN);

 else if (digitalRead(BUTTON\_BLUE) == 0) return(CHOICE\_BLUE);

 else if (digitalRead(BUTTON\_YELLOW) == 0) return(CHOICE\_YELLOW);

 return(CHOICE\_NONE); // If no button is pressed, return none

}

// Light an LED and play tone

// Red, upper left: 440Hz - 2.272ms - 1.136ms pulse

// Green, upper right: 880Hz - 1.136ms - 0.568ms pulse

// Blue, lower left: 587.33Hz - 1.702ms - 0.851ms pulse

// Yellow, lower right: 784Hz - 1.276ms - 0.638ms pulse

void toner(byte which, int buzz\_length\_ms)

{

 setLEDs(which); //Turn on a given LED

 //Play the sound associated with the given LED

 switch(which)

 {

 case CHOICE\_RED:

 buzz\_sound(buzz\_length\_ms, 1136);

 break;

 case CHOICE\_GREEN:

 buzz\_sound(buzz\_length\_ms, 568);

 break;

 case CHOICE\_BLUE:

 buzz\_sound(buzz\_length\_ms, 851);

 break;

 case CHOICE\_YELLOW:

 buzz\_sound(buzz\_length\_ms, 638);

 break;

 }

 setLEDs(CHOICE\_OFF); // Turn off all LEDs

}

// Toggle buzzer every buzz\_delay\_us, for a duration of buzz\_length\_ms.

void buzz\_sound(int buzz\_length\_ms, int buzz\_delay\_us)

{

 // Convert total play time from milliseconds to microseconds

 long buzz\_length\_us = buzz\_length\_ms \* (long)1000;

 // Loop until the remaining play time is less than a single buzz\_delay\_us

 while (buzz\_length\_us > (buzz\_delay\_us \* 2))

 {

 buzz\_length\_us -= buzz\_delay\_us \* 2; //Decrease the remaining play time

 // Toggle the buzzer at various speeds

 digitalWrite(BUZZER1, LOW);

 digitalWrite(BUZZER2, HIGH);

 delayMicroseconds(buzz\_delay\_us);

 digitalWrite(BUZZER1, HIGH);

 digitalWrite(BUZZER2, LOW);

 delayMicroseconds(buzz\_delay\_us);

 }

}

// Play the winner sound and lights

void play\_winner(void)

{

 setLEDs(CHOICE\_GREEN | CHOICE\_BLUE);

 winner\_sound();

 setLEDs(CHOICE\_RED | CHOICE\_YELLOW);

 winner\_sound();

 setLEDs(CHOICE\_GREEN | CHOICE\_BLUE);

 winner\_sound();

 setLEDs(CHOICE\_RED | CHOICE\_YELLOW);

 winner\_sound();

}

// Play the winner sound

// This is just a unique (annoying) sound we came up with, there is no magic to it

void winner\_sound(void)

{

 // Toggle the buzzer at various speeds

 for (byte x = 250 ; x > 70 ; x--)

 {

 for (byte y = 0 ; y < 3 ; y++)

 {

 digitalWrite(BUZZER2, HIGH);

 digitalWrite(BUZZER1, LOW);

 delayMicroseconds(x);

 digitalWrite(BUZZER2, LOW);

 digitalWrite(BUZZER1, HIGH);

 delayMicroseconds(x);

 }

 }

}

// Play the loser sound/lights

void play\_loser(void)

{

 setLEDs(CHOICE\_RED | CHOICE\_GREEN);

 buzz\_sound(255, 1500);

 setLEDs(CHOICE\_BLUE | CHOICE\_YELLOW);

 buzz\_sound(255, 1500);

 setLEDs(CHOICE\_RED | CHOICE\_GREEN);

 buzz\_sound(255, 1500);

 setLEDs(CHOICE\_BLUE | CHOICE\_YELLOW);

 buzz\_sound(255, 1500);

}

// Show an "attract mode" display while waiting for user to press button.

void attractMode(void)

{

 while(1)

 {

 setLEDs(CHOICE\_RED);

 delay(100);

 if (checkButton() != CHOICE\_NONE) return;

 setLEDs(CHOICE\_BLUE);

 delay(100);

 if (checkButton() != CHOICE\_NONE) return;

 setLEDs(CHOICE\_GREEN);

 delay(100);

 if (checkButton() != CHOICE\_NONE) return;

 setLEDs(CHOICE\_YELLOW);

 delay(100);

 if (checkButton() != CHOICE\_NONE) return;

 }

}

//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

// The following functions are related to Beegees Easter Egg only

// Notes in the melody. Each note is about an 1/8th note, "0"s are rests.

int melody[] = {

 NOTE\_G4, NOTE\_A4, 0, NOTE\_C5, 0, 0, NOTE\_G4, 0, 0, 0,

 NOTE\_E4, 0, NOTE\_D4, NOTE\_E4, NOTE\_G4, 0,

 NOTE\_D4, NOTE\_E4, 0, NOTE\_G4, 0, 0,

 NOTE\_D4, 0, NOTE\_E4, 0, NOTE\_G4, 0, NOTE\_A4, 0, NOTE\_C5, 0};

int noteDuration = 115; // This essentially sets the tempo, 115 is just about right for a disco groove :)

int LEDnumber = 0; // Keeps track of which LED we are on during the beegees loop

// Do nothing but play bad beegees music

// This function is activated when user holds bottom right button during power up

void play\_beegees()

{

 //Turn on the bottom right (yellow) LED

 setLEDs(CHOICE\_YELLOW);

 toner(CHOICE\_YELLOW, 150);

 setLEDs(CHOICE\_RED | CHOICE\_GREEN | CHOICE\_BLUE); // Turn on the other LEDs until you release button

 while(checkButton() != CHOICE\_NONE) ; // Wait for user to stop pressing button

 setLEDs(CHOICE\_NONE); // Turn off LEDs

 delay(1000); // Wait a second before playing song

 digitalWrite(BUZZER1, LOW); // setup the "BUZZER1" side of the buzzer to stay low, while we play the tone on the other pin.

 while(checkButton() == CHOICE\_NONE) //Play song until you press a button

 {

 // iterate over the notes of the melody:

 for (int thisNote = 0; thisNote < 32; thisNote++) {

 changeLED();

 tone(BUZZER2, melody[thisNote],noteDuration);

 // to distinguish the notes, set a minimum time between them.

 // the note's duration + 30% seems to work well:

 int pauseBetweenNotes = noteDuration \* 1.30;

 delay(pauseBetweenNotes);

 // stop the tone playing:

 noTone(BUZZER2);

 }

 }

}

// Each time this function is called the board moves to the next LED

void changeLED(void)

{

 setLEDs(1 << LEDnumber); // Change the LED

 LEDnumber++; // Goto the next LED

 if(LEDnumber > 3) LEDnumber = 0; // Wrap the counter if needed

}

# Exercise 32, Morse Code transmitter

// sketch 5-05

const int ledPin = 13;

const int dotDelay = 200;

char\* letters[] = {

 ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", // A-I

 ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", // J-R

 "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.." // S-Z

};

char\* numbers[] = {

 "-----", ".----", "..---", "...--", "....-", ".....", "-....", "--...", "---..", "----."};

void setup()

{

 pinMode(ledPin, OUTPUT);

 Serial.begin(9600);

}

void loop()

{

 char ch;

 if (Serial.available() > 0)

 {

 ch = Serial.read();

 if (ch >= 'a' && ch <= 'z')

 {

 flashSequence(letters[ch - 'a']);

 }

 else if (ch >= 'A' && ch <= 'Z')

 {

 flashSequence(letters[ch - 'A']);

 }

 else if (ch >= '0' && ch <= '9')

 {

 flashSequence(numbers[ch - '0']);

 }

 else if (ch == ' ')

 {

 delay(dotDelay \* 4); // gap between words

 }

 }

}

void flashSequence(char\* sequence)

{

 int i = 0;

 while (sequence[i] != NULL)

 {

 flashDotOrDash(sequence[i]);

 i++;

 }

 delay(dotDelay \* 3); // gap between letters

}

void flashDotOrDash(char dotOrDash)

{

 digitalWrite(ledPin, HIGH);

 if (dotOrDash == '.')

 {

 delay(dotDelay);

 }

 else // must be a dash

 {

 delay(dotDelay \* 3);

 }

 digitalWrite(ledPin, LOW);

 delay(dotDelay); // gap between flashes

}