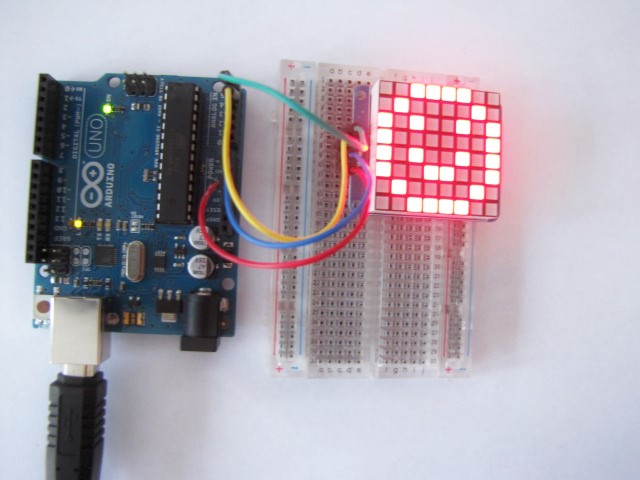
Computer Science

Light Lab

Teacher Guide

Suzanne Hanbicki

6-12 Business & Computer Science Education Curriculum Leader

Downingtown Area School District

Downingtown, Pennsylvania

Gavin Speirs

6-12 Technology Education Curriculum Leader

Downingtown Area School District

Downingtown, Pennsylvania

**Drexel University Mentor:**

Dr. Dario Salvucci

Associate Dean of CCI Undergraduate Studies

Professor of Computer Science

Drexel University

Philadelphia, Pennsylvania

****

**REThink@Drexel**

**Research Experiences for Teachers in Engineering and Computer Science:** **Machine Learning, Big Data and CS Principles**

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1301171.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Contents

[Light Lab Teacher’s Guide 3](#_Toc413872229)

[Overview - Microprocessors 3](#_Toc413872230)

[Learning Objectives 3](#_Toc413872231)

[Prerequisites 4](#_Toc413872232)

[What is Provided 4](#_Toc413872233)

[Installation/Setup 4](#_Toc413872234)

[Hardware Components and set up 5](#_Toc413872235)

[Student Activities 6](#_Toc413872236)

[Activity 1: Binary conversion 6](#_Toc413872237)

[Activity 2: Circuit Basics and Prototype Build 6](#_Toc413872238)

[Activity 3: Software Sketch 7](#_Toc413872239)

[Light Lab 9](#_Toc413872240)

[Microcontrollers 9](#_Toc413872241)

[Integrated Development Environment 9](#_Toc413872242)

[Hardware Components 9](#_Toc413872243)

[Student Activities 10](#_Toc413872244)

[Binary Code 10](#_Toc413872245)

[**Light Matrix Project** 11](#_Toc413872246)

[Circuit Basics 11](#_Toc413872247)

[Create your prototype 12](#_Toc413872248)

[Program your Arduino prototype 13](#_Toc413872249)

[Create your own 8-bit art 14](#_Toc413872250)

[Optional Activity 14](#_Toc413872251)

[Appendix A 15](#_Toc413872252)

[Appendix B 16](#_Toc413872253)

[Appendix C 17](#_Toc413872254)

[Appendix D 18](#_Toc413872255)

[Appendix E 19](#_Toc413872256)

[Appendix F 20](#_Toc413872257)

[References 21](#_Toc413872258)

# Light Lab Teacher’s Guide

# Overview - Microprocessors

This module uses microprocessors to assist students understanding of computer science concepts. It shows students how ubiquitous computers are and how computational thinking can be used to solve many problems. Implementation of this lab can occur within a classroom over several class periods or can be shortened for a computer science workshop activity. Microprocessors can be very powerful when used in cross-curricular projects bringing students from technology education, life sciences, math and family and consumer science to the world of computer science.

With the current need to introduce more students of varying backgrounds to computer science, and to provide a richer experience to engage these students, the Light Lab can assist in the students’ understanding of the coding concepts and may aid in the retention of these students in a computer science program.

The Light Lab allows students to visualize the code in a program using a bi-color eight by eight LED matrix. The visualization will support the understanding of binary code, base two to base ten conversion, loops, and control statements with the added bonus of the excitement of LED lighting. With slight changes in the length and depth of the project, this lab can be used at elementary, middle school or high school levels. With the use of base ten concepts in the Common Core Standards, this lab can underscore the number systems concepts in accelerated elementary math classes. It allows students of introductory or exploring computer science courses to explore both hardware and software. It can also help integrate software concepts into a technology education course at the middle school level. This lab also gives students in the AP Computer Science course an interesting way to learn binary to decimal conversion. It received great feedback when used during a middle school technology workshop for girls.

The Light Lab is provided through Drexel University Research Experiences for Teachers in Engineering and Computer Science Site for Big Data, Machine Learning and CS Principles. This module is based upon the work of Dr. Dario Salvucci, Associate Dean of CCI Undergraduate Studies and Professor of Computer Science at Drexel University. His research focuses on understanding human multitasking, and most of his work involves computational cognitive models to represent and simulate thoughts and behaviors. He developed the Distract-R program to study the effects of distraction on driver reaction. The Distract-R program uses a graphic simulation to show the effects of distraction while using electronic devices when driving and allows “a designer to prototype basic interfaces, demonstrate possible tasks on these interfaces, specify relevant driver characteristics and driving scenarios, and finally simulate, visualize, and analyze the resulting behavior as generated by the cognitive model” (Salvucci, 2011).

One cannot simply engage students by using flashing lights, although they do a pretty good job on their own. The concepts covered also need to be relevant to the student and connected to real world situations. The Light Lab begins with a discussion on visualization of code and visual stimulation and connects computing to their physical world. Further work with Arduino projects will advance problem-solving skills and allow students to develop solutions to real-world problems.

## Learning Objectives

Given the elements studied, the lights lab is suitable for use at the beginning of an introductory computer science course or in AP Computer Science when discussing binary code. When students have a clear understanding of the basics of programming including algorithm development, usage of variables and are ready to learn for loops and condition statements, additional projects can use the same prototype. This will give students a visual representation of their programs. This can also be used early in an AP Computer Science course to introduce or review loops and condition statements.

## Prerequisites

Students should be able to use methods and be quite familiar with an Integrated Development Environment (IDE). Students should also be able to locate files and copy them to the correct location in the IDE.

## What is Provided

Student instruction files and solution files are available. Text for the Arduino code is available in the Appendix.

## Installation/Setup

You will need the Arduino IDE, which is a free download available from the Arduino website. Two libraries are also necessary for the Sketches to run the programs to control the LEDs. You will also need the hardware components, which are fairly inexpensive and robust, and can lead to further exploration of microcontrollers and problem-solving.

Download the Arduino IDE from the Arduino website. There is a ZIP file for a non-administrator installation. This can be downloaded and extracted to the desktop. Open the Arduino IDE, and an Arduino folder will be placed in your Documents folder. This will include a Libraries folder. Download the Adafruit® GFX library and Adafruit LED Backpack libraries available from the Adafruit web site, which will have a link to the files on GIThub. After extracting these files to your Libraries folder within the Arduino folder, you will need to rename the library folder names to change the dashes to underscores. You will need to close and reopen the Arduino IDE when finished installing the libraries.

As of this writing, there is a conflict between files in the Robot Control library and the GFX library, which can be fixed easily by deleting the Robot Control library.

If you envision continuing with additional Arduino activities or assigning students a build project that may include a smaller microprocessor such as the Trinket, Gemma or Flora, you should install an Adafruit version of the Arduino IDE. This can be found at: https://learn.adafruit.com/introducing-gemma/setting-up-with-arduino-ide.

At the time of this writing, if you are using Windows 8, you do not need to specifically install drivers for Arduino. You will need to do so in earlier versions of Windows. Use a USB to USB-B cable to connect the Arduino to your computer. Open the Control Panel, and within System and Security, open the Device Manager. This can also be found under Devices and Printers. Find and right-click the Arduino in the list and choose Properties, and then use the Update Driver button under the Driver tab. Follow the prompts and browse to the install location of Arduino and select the Drivers folder within. Windows will then find and install the drivers. Once you set up your prototype, you will now be able to upload your Sketches. As this is open source, the developers are continually changing and updating the source code and the hardware. Check the Arduino forums if you need additional help uploading a specific board or installing libraries.

We recommend loading the TwoDot sketch onto each Arduino prior to the project. When students complete their circuit and power their prototype, having a program installed will give them a visual cue that they wired their circuit correctly. Code for each sketch is in the Appendix.

You will need the listed hardware components. Paired programming will allow students to learn with each other and will decrease total cost for a classroom set. The price is half that of an introductory robot and building the project as a prototype will allow reuse of all of the components.

## Hardware Components and set up

You will need the following components:

**Adafruit Bicolor LED Square Pixel Matrix with I2C Backpack $15.95**

https://www.adafruit.com/search?q=bicolor+LED

(Single color matrices are also available, which are less expensive)

**Half Sized Breadboard $5.00**

https://www.adafruit.com/products/64

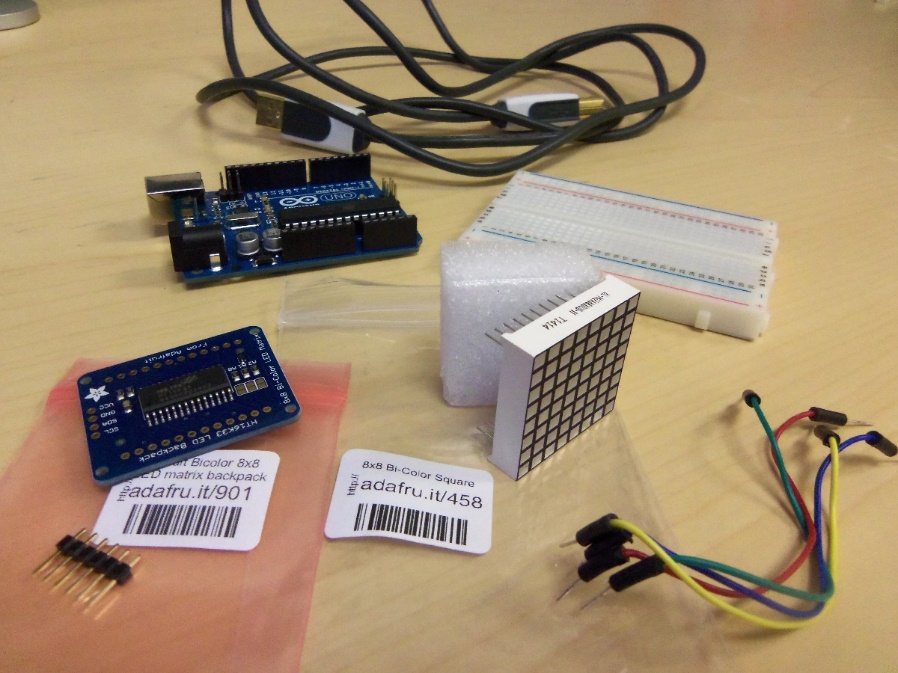
**Breadboarding wire bundle $6.00**

https://www.adafruit.com/search?q=breadboarding+wire+bundle

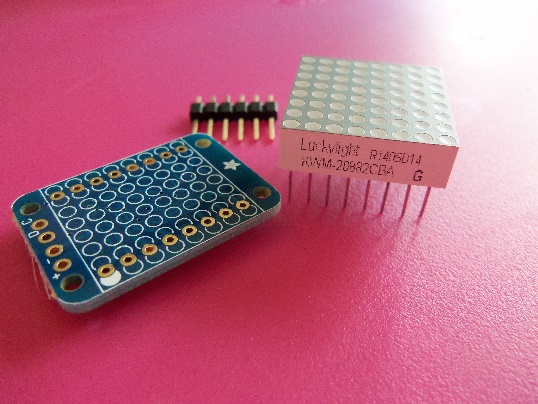
**Arduino UNO R3 $24.95**

https://www.adafruit.com/products/50

**Standard USB A-B cable $3.95**

https://www.adafruit.com/products/62

It would be best to complete the hardware portion of this cross-curricular project in an electronics lab. If that is not available to you, a responsible adult should solder the backpack and 4-pin header. Adafruit has a soldering guide for those that are new to electronics. Although conductive glue is available on the market, we found that difficult to use and too easy to overflow the joint and produce short circuits.

It is essential to position the matrix correctly onto the backpack prior to soldering. The screen-printed side of the backpack should be toward the LED matrix and the side of the LED matrix with printing should line up with the filled screen-printed dot on the backpack. The pins should slide right into the holes.

If you position this incorrectly, it is quite difficult to fix, so ensure you align it correctly! Flip the two parts over and lay the module on the lights. This way you can access the pins to solder them. Solder and trim the 24 pins. Finally, solder the 4-pin header to the backpack.

## Student Activities

### Activity 1: Binary conversion

We recommend starting the lab with a discussion on binary (base-2) and how to convert binary to decimal (base 10) using a conversion table. For advanced students or courses, conversion to hexadecimal is a logical next step. An optional handout for students to practice conversion is in Appendix A. If you have time, we recommend using the CSUnplugged dots activity (Bell et al., 2015, p. 3) for a kinesthetic way to show the algorithm of binary counting and to further solidify the understanding of binary.

Adafruit has a good video on decimal, binary and hexadecimal conversion on their YouTube© channel that could help reinforce the conversion concepts.

### Activity 2: Circuit Basics and Prototype Build

An electrical circuit powers the microcontroller and components. A complete circuit is needed in order for the power to flow from the source, through the project and back. This circuit will also carry the programming information. The students will prototype the circuit using an Arduino Uno R3, a breadboard, jump wires, a battery with a barrel connector and a bi-color 8x8 LED matrix. Further exploration of computer hardware can be accomplished by using a project similar to the computer-purchasing project within the Exploring Computer Science curriculum (Goode and Chapman, 2013, p. 18). The prototype that the students will build along with the Sketch they will upload will solidify binary concepts and allow students to see the interaction of hardware and software.

If you need background material in order to introduce electricity basics to the students, there are many electricity and circuit videos on YouTube or TeacherTube, including a wonderful episode of Bill Nye the Science Guy on electronics and circuits.

The circuit will make an electrical connection from pins on the Aruduino Uno R3 to the header pins on the matrix backpack and back to the Arduino.

### Activity 3: Software Sketch

Once students have the prototype built, show the smile sketch. To underscore the binary lesson, show the array code with the ones highlighted as below:

{ B 0 0 1 1 1 1 0 0,

B 0 1 0 0 0 0 1 0,

B 1 0 1 0 0 1 0 1,

B 1 0 0 0 0 0 0 1,

B 1 0 1 0 0 1 0 1,

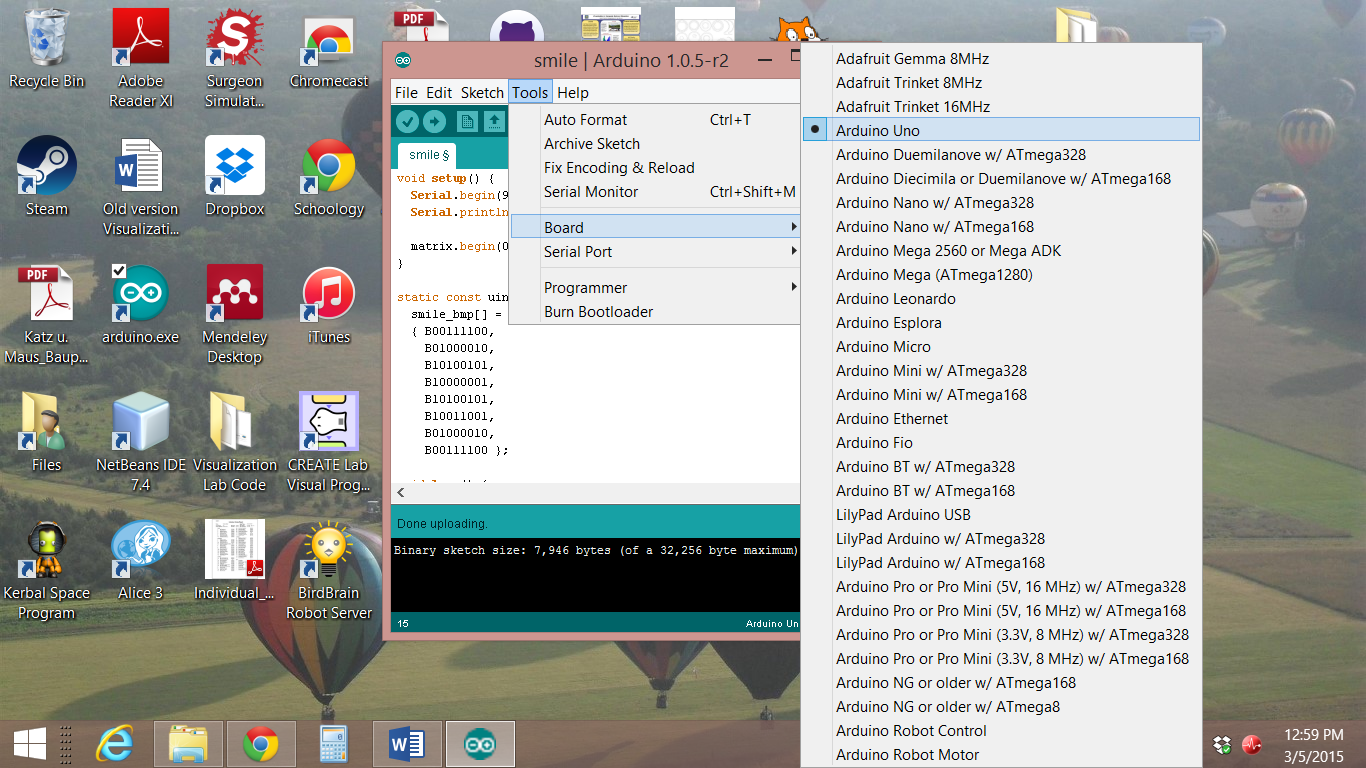
B 1 0 0 1 1 0 0 1,

B 0 1 0 0 0 0 1 0,

B 0 0 1 1 1 1 0 0 };

The ones in the array will turn on the lights and the zeros will turn the lights off.

Specify the correct microcontroller by selecting the Uno within the Tools menu.



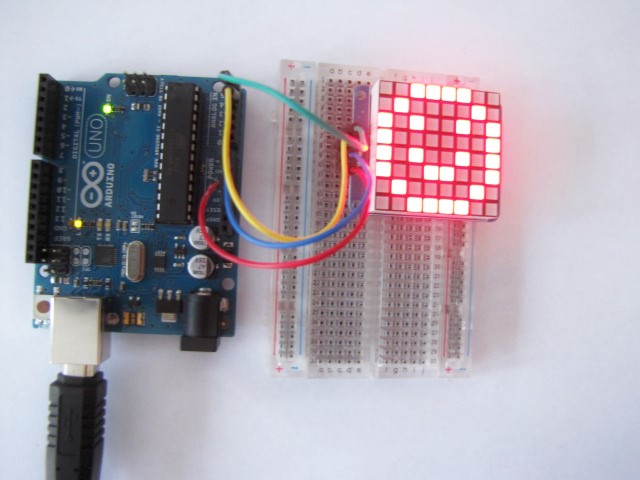
Give students the opportunity to change the code to show their own individual 8-bit art on the prototype.

These prototypes can support lessons in loops and methods and present the students with a wonderful visualization of their code. An additional sketch is available in the Appendix that can scroll text on the matrix and will allow the visualization of more complex programming concepts.

Computer Science

Light Lab

Student Guide



Suzanne Hanbicki

6-12 Business and Computer Science Education Curriculum Leader

Downingtown Area School District

Downingtown, Pennsylvania

Gavin Speirs

6-12 Technology Education Curriculum Leader

Downingtown Area School District

Downingtown, Pennsylvania

**Drexel University Mentor:**

Dr. Dario Salvucci

Associate Dean of CCI Undergraduate Studies

Professor of Computer Science

Drexel University

Philadelphia, Pennsylvania

****

**REThink@Drexel**

**Research Experiences for Teachers in Engineering and Computer Science:** **Machine Learning, Big Data and CS Principles**

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1301171.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

# Light Lab

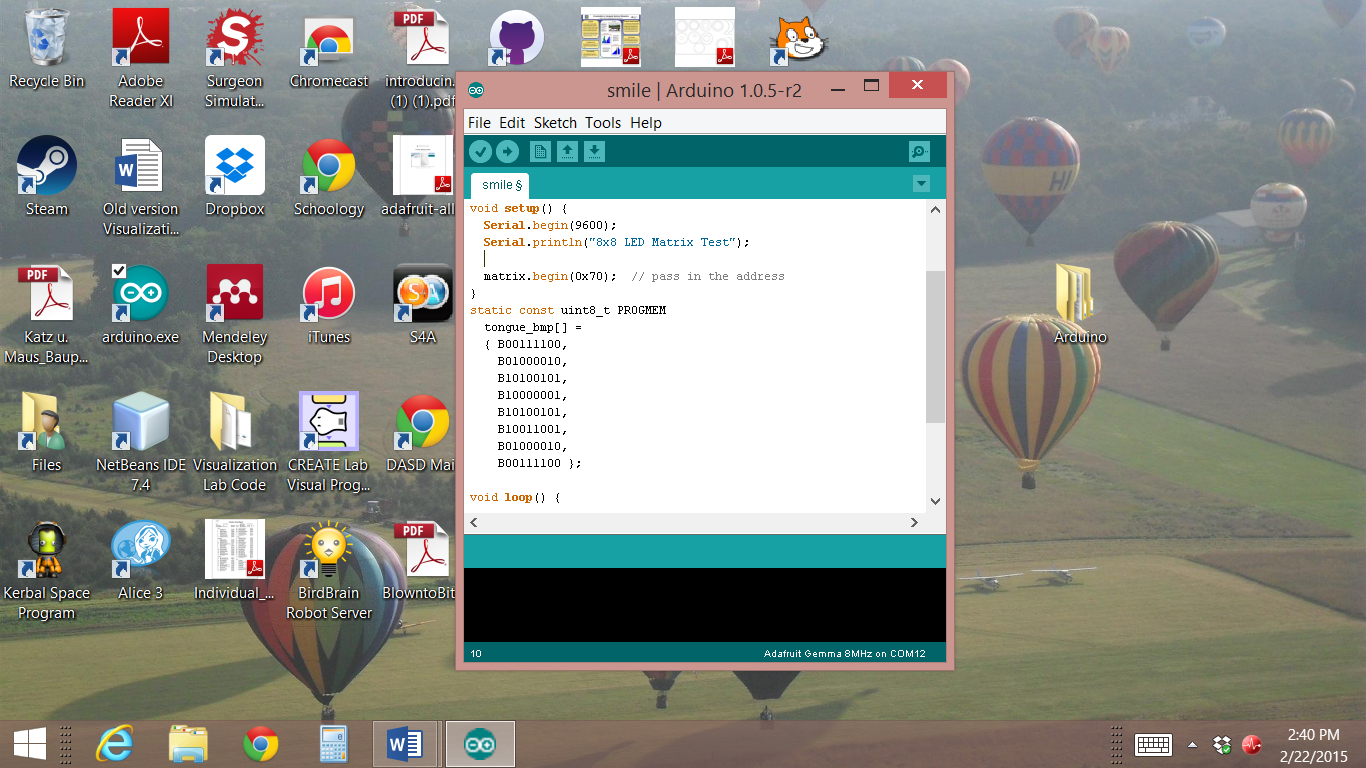
You will learn fundamental introductory computer science concepts along with basic computer hardware, circuitry and electronics in this lab. You will prototype a circuit using a LED light matrix and a microcontroller to show your understanding of binary numbers.

## Microcontrollers

Microcontrollers are simply small computers. They have a single integrated circuit board, analog and digital pins, a USB interface which can serve both as power and enable you to upload your program, and a power jack for use independent from your computer. Microcontrollers come in many sizes and shapes and can be used on many different projects. E-textiles (necklaces, hats, and jackets), automated assistance (sprinkler systems, door locks) and detection systems (noxious fume alerts, airflow analysis) are just some examples of application projects.

Microcontrollers such as Arduino and Raspberry Pi are open-source electronics prototyping platforms. Open-source means that the resources for the hardware and software are freely available to be used, changed and redistributed. A quick browser search with the keywords Arduino or Raspberry Pi will find many projects with very detailed instructions.

## Integrated Development Environment

You use an integrated development environment (IDE) to write programs, called Sketches, for the Arduino. The Arduino uses the IDE pictured below. The language is fairly simple, yet robust enough to use for the project you envision. Each sketch needs two elements, a setup and a loop.

The setup portion of the sketch contains what amounts to the basic information and variables needed. The loop contains the action parts of the program. Whatever problem you are trying to solve, be it sensing, lighting, alerting or tweeting, we want it to happen continually. As we want our microcontroller to run the sketch the entire time our project is turned on, all of the action parts of the program reside within the loop.

## Hardware Components

This lab uses an 8x8 LED matrix to show the algorithm of the code, which produces the light patterns.

You will build a prototype circuit using an Arduino Uno R3, a breadboard, jump wires, a battery with a barrel connector and a bi-color 8x8 LED matrix. You will then use an IDE and your computer to upload a Sketch onto your prototype. You can then change the code to show the 8-bit art of your own making.

# Student Activities

## Binary Code

Computers do not actually understand words. Computers are electrical devices; they only understand series of 0s and 1s. Electricity is either on or off which is represented with 1s, which are on, and 0s, which are off. This is called binary code or base-2. Every interaction with a computer inevitably is taken down to the binary level. Binary simply means that something is on or it is off. So the binary code says to turn individual bits on, or off.

It is not very easy to write a program using only 0s and 1s. Binary code represents text or programming instructions using the binary code system. It is easy to convert binary to decimal or base-10 using a conversion table.

If we compare base-2 to base-10, it is easier to understand. Base-10 number 15 is made up of five ones and one ten. This is represented in the table below. Each column in the table is ten times larger than the one on its right.

|  |  |
| --- | --- |
| 10 | 1 |
| 1 | 5 |

Binary only uses 1s and 0s. Which means that each column on the table is two times larger than the one on its right. Each 0 or 1 is a bit. To represent the number 15, we would need to use four bits. When a bit is on, it is equal to the value in that column. You would look at what bits are on, add the values together and that is the value of that number in decimal.

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 4 | 2 | 1 |
| 1 | 1 | 1 | 1 |

So 8 + 4 + 2 + 1 = 15. The number 1111 in binary is equal to 15 in decimal.

What would 0101 equal?

What is 71 in binary?

Your teacher may ask that you complete more binary to decimal conversions. These can be found in Appendix A.

For the most part, we do not have to worry about binary code, as the computer processes the input automatically into the binary language it understands. We do need to know that binary is the building block, and the bits are either on, meaning a 1, or off, with a 0.

## **Light Matrix Project**

## Circuit Basics

You will need to have a complete circuit in order for the power to flow from the source, through the project and back. The electrical circuit powers the microcontroller and components. This circuit will also carry the programming information. You will put together the prototype using an Arduino Uno R3, a breadboard, jump wires, a battery with a barrel connector and a bi-color 8x8 LED matrix.

You can assemble and program this 8x8 square of LEDs to show an 8-bit graphic or use it to scroll shapes and text. We will first work with the graphic in the form of a smiley face.

You will need the following components:

**Adafruit Bicolor LED Square Pixel Matrix with I2C Backpack**

https://www.adafruit.com/search?q=bicolor+LED

**Half Sized Breadboard**

https://www.adafruit.com/products/64

**Breadboarding wire bundle**

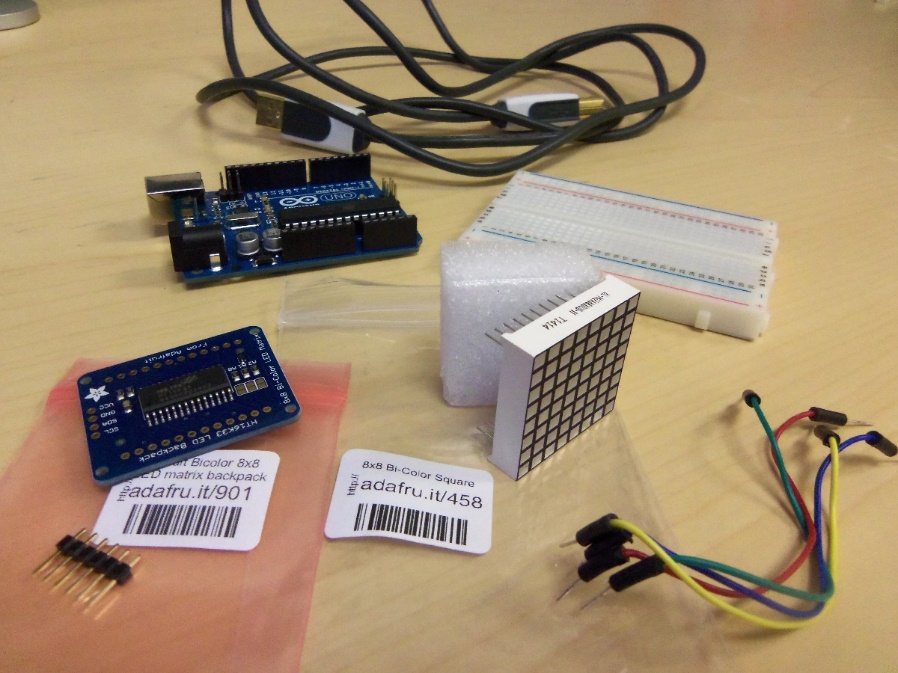
https://www.adafruit.com/search?q=breadboarding+wire+bundle

**Arduino UNO R3**

https://www.adafruit.com/products/50

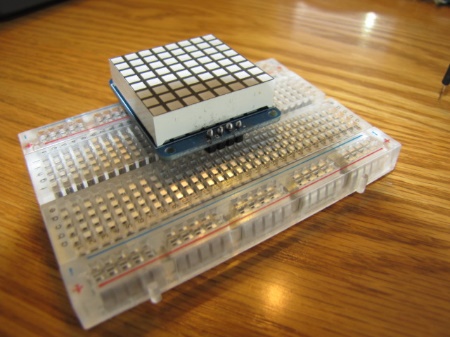
**Standard USB A-B cable**

https://www.adafruit.com/products/62



## Create your prototype

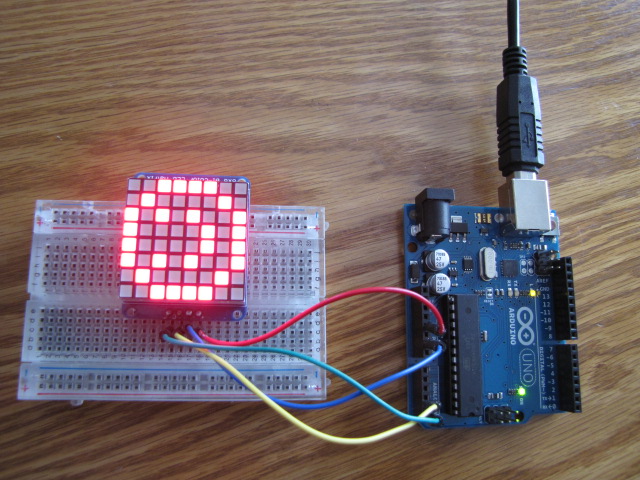
The breadboard is a great tool to use for prototyping. You could solder wires directly to the backpack, but putting this circuit together as a prototype allows you to use the components for many different projects. You will be able to change individual components easily.

The breadboard itself is composed of a rectangle of plastic with rows of holes. These rows are called terminal strips. Under the holes are strips of metal that connect the rows. When you push a wire into one hole on the board and another into a hole in the same row, the metal strip will connect the two parts of your circuit. Most breadboards are divided into two halves. The rows on each side of the middle divider are not electrically connected. If you plug the component in one side and the jump wire into the other, you will need a jump wire to bridge the gap and connect the circuit.

Gently push the 8x8 matrix with the header pins attached into your breadboard, leaving enough room to access the breadboard holes in the same rows. It does not matter where you plug the matrix in, but you should pick a spot so that the component is flat, so as not to stress any of the solder joints.

Then insert the jump wires into position in the same rows as the header pins. Jump wires come in a variety of colors so that you can trace the circuit easily. It does not matter which colors you use. Electronic conventions are black for ground and red for power. On our prototype, we used blue for ground. You should always plug the ground wire in first. It is important that you do not power your circuit without the ground attached.

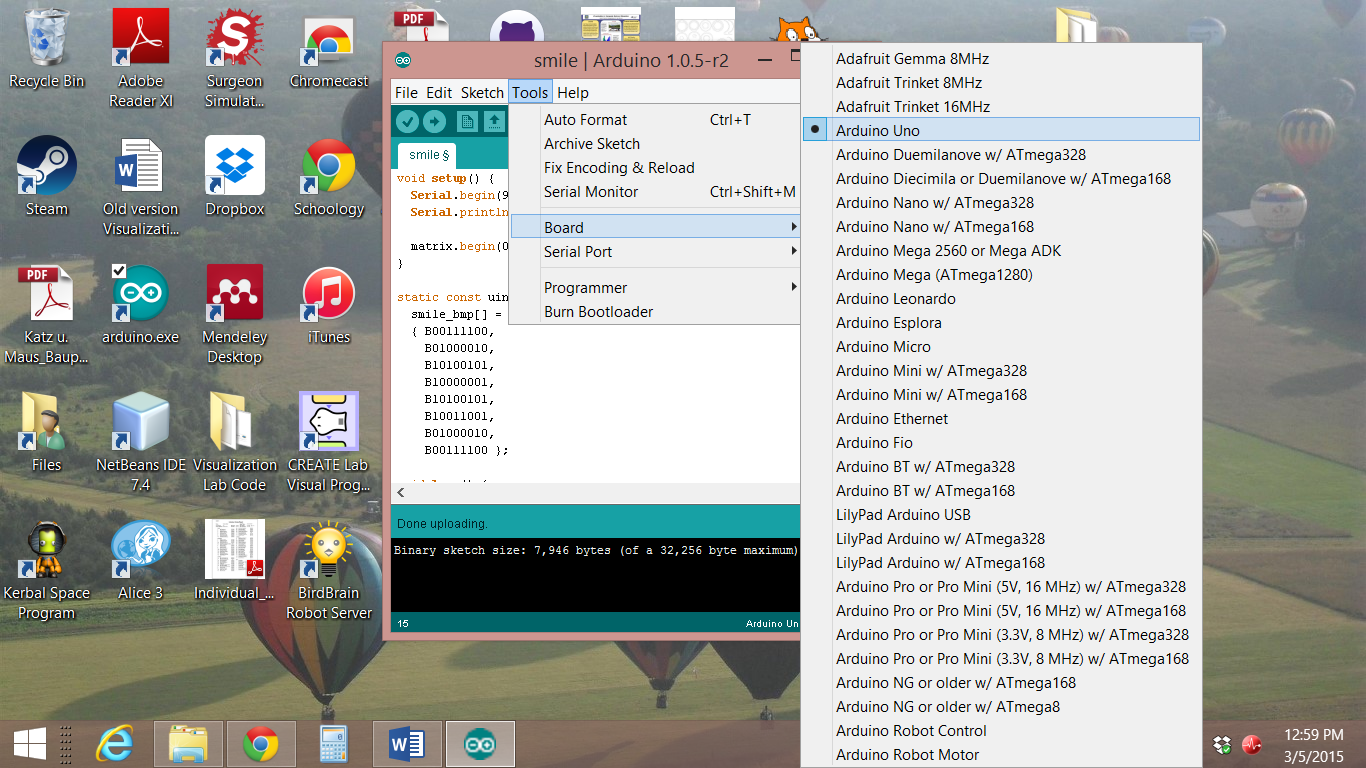
Look closely at the top of your matrix component. There are labels for each of the header pins. These are +, -, D and C. + stands for power in, - is ground. Look closely at the Arduino board. You will see General Purpose Input/Output pins with white labels. The opposite ends of the jump wires for the header pins will go into specific pins on the Arduino. Start with the jump wire which is plugged into the –. Push the other end of this jump wire into one of the two GND pins on the side labeled POWER. Connect the D to Analog #4 (A4), and connect the C into Analog #5 (A5). Then connect the + to 5V. Power for your circuit can come from the battery pack or through the USB cable. You will need the cable to upload the sketch. If your circuit is correct, and the reset sketch previously was uploaded to your Arduino, you will see two lights in the middle of your matrix.

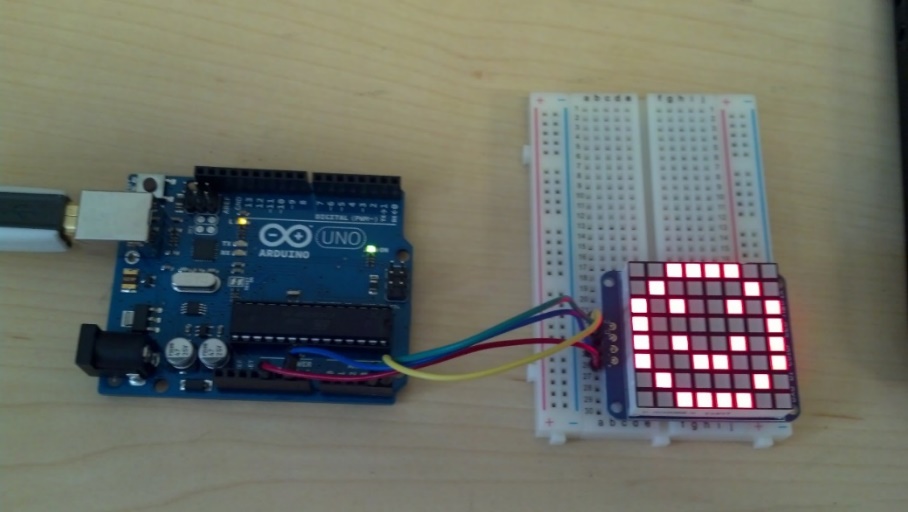


Connect the jump wires in the following manner:

* + connects to 5v
* - connects to GND
* C connects to 5A
* D connects to 4A

## Program your Arduino prototype

Open the Arduino IDE. Copy the text for the Smile sketch in the Appendix into a blank sketch. Ensure you have the correct board specified under Tools by selecting Board, then Arduino Uno. The check mark at the top compiles your code and the arrow uploads to your Arduino.

 When you successfully upload your sketch, it will smile at you!

## Create your own 8-bit art

While you have the smile sketch open in the IDE, scroll down in the code until you find the smile\_bmp array. This is a collection of eight bytes of 0 and 1 bits. This is what is turning on the individual LEDs in the matrix. The square brackets tell the compiler that the code within the curly braces is a collection of items.

smile\_bmp[] =

{ B00111100,

B01000010,

B10100101,

B10000001,

B10100101,

B10011001,

B01000010,

B00111100 };

You can see how the ones are on in the matrix and the zeroes are off.

You can use the grid in the Appendix to draw your own 8-bit art. Once you have a drawing, change the code within the array and upload it to your prototype to share your art with everyone.

## Optional Activity

You can do much more with the program and the prototype. You can use a count-controlled loop, called a for loop, to scroll text on the matrix. Within the code below, the text after the // is a comment. The compiler ignores the text after it sees the double forward slash and starts paying attention again when it gets to the next line. These are used as notes to yourself and other programmers. If you use this code, you do not need to type the comments - these just explain what each line of the code does.

{

matrix.setTextWrap(false); // we don't want text to wrap so it scrolls nicely

matrix.setTextSize(1); //this sets the text size

matrix.setTextColor(LED\_GREEN); //this sets the color of the LEDs

for (int8\_t x=7; x>=-36; x--) { //this is the for loop which controls the number of loops

matrix.clear(); // clears the last art from the matrix

matrix.setCursor(x,0); //places the cursor at the top left

matrix.print("REThink"); //whatever is within the quotation marks will show

matrix.writeDisplay(); //tells the compiler to go ahead and show the text

delay(100); //sets a pause so that we can read the display

}

You can copy the whole sketch from the appendix and manipulate it to write a scrolling message to the world.

# Appendix A

**Converting Binary to Decimal**

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20**

Fill in the missing powers of 2 in the series below:

1, 2, 4, 8, \_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, 4096

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4096 |  |  |  |  |  |  |  |  | 8 | 4 | 2 | 1 |

Converting Binary numbers to decimal:

Example: 1101 = 13

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 4 | 2 | 1 |
| 1 | 1 | 0 | 1 |
| 8 | 4 | 0 | 1 |

1 x 8 = 8

1 x 4 = 4

0 x 2 = 0

1 x 1 = 1

13

Convert the following numbers from binary to decimal:

0000 \_\_\_\_\_\_\_\_\_\_

0001 \_\_\_\_\_\_\_\_\_\_

0111 \_\_\_\_\_\_\_\_\_\_

1000 \_\_\_\_\_\_\_\_\_\_

1110 \_\_\_\_\_\_\_\_\_\_

1111 \_\_\_\_\_\_\_\_\_\_

# Appendix B

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

//Original code courtesy of Adafruit.com, where you can purchase all of the hardware items needed for //this project. The code below has been amended for use with the binary lesson plan

This is a library for our I2C LED Backpacks

Designed specifically to work with the Adafruit LED Matrix backpacks

----> http://www.adafruit.com/products/872

----> http://www.adafruit.com/products/871

----> http://www.adafruit.com/products/870

These displays use I2C to communicate, 2 pins are required to interface. There are multiple selectable I2C addresses. For backpacks with 2 Address Select pins: 0x70, 0x71, 0x72 or 0x73. For backpacks with 3 Address Select pins: 0x70 thru 0x77

Adafruit invests time and resources providing this open source code, please support Adafruit and open-source hardware by purchasing products from Adafruit!

Written by Limor Fried/Ladyada for Adafruit Industries.

BSD license, all text above must be included in any redistribution

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

#include <Wire.h>

#include "Adafruit\_LEDBackpack.h"

#include "Adafruit\_GFX.h"

Adafruit\_BicolorMatrix matrix = Adafruit\_BicolorMatrix();

void setup() {

Serial.begin(9600);

Serial.println("8x8 LED Matrix Test");

matrix.begin(0x70); // pass in the address

}

static const uint8\_t PROGMEM

twodot\_bmp[] =

{ B00000000,

B00000000,

B00000000,

B00000000,

B00110000,

B00000000,

B00000000,

B00000000 };

void loop() {

matrix.clear();

matrix.drawBitmap(0, 0, twodot\_bmp, 8, 8, LED\_RED);

matrix.writeDisplay();

delay(500);

}

# Appendix C

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

//Original code courtesy of Adafruit.com, where you can purchase all of the hardware items needed for //this project The code below has been amended for use with the binary lesson plan

This is a library for our I2C LED Backpacks

Designed specifically to work with the Adafruit LED Matrix backpacks

----> http://www.adafruit.com/products/872

----> http://www.adafruit.com/products/871

----> http://www.adafruit.com/products/870

These displays use I2C to communicate, 2 pins are required to interface. There are multiple selectable I2C addresses. For backpacks with 2 Address Select pins: 0x70, 0x71, 0x72 or 0x73. For backpacks with 3 Address Select pins: 0x70 thru 0x77

Adafruit invests time and resources providing this open source code, please support Adafruit and open-source hardware by purchasing products from Adafruit!

Written by Limor Fried/Ladyada for Adafruit Industries.

BSD license, all text above must be included in any redistribution

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

#include <Wire.h>

#include "Adafruit\_LEDBackpack.h"

#include "Adafruit\_GFX.h"

Adafruit\_BicolorMatrix matrix = Adafruit\_BicolorMatrix();

void setup() {

Serial.begin(9600);

Serial.println("8x8 LED Matrix Test");

matrix.begin(0x70); // pass in the address

}

static const uint8\_t PROGMEM

smile\_bmp[] =

{ B00111100,

B01000010,

B10100101,

B10000001,

B10100101,

B10011001,

B01000010,

B00111100 };

void loop() {

matrix.clear();

matrix.drawBitmap(0, 0, smile\_bmp, 8, 8, LED\_RED);

matrix.writeDisplay();

delay(500);

}

# Appendix D

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

This is a library for our I2C LED Backpacks

Designed specifically to work with the Adafruit LED Matrix backpacks

----> http://www.adafruit.com/products/872

----> http://www.adafruit.com/products/871

----> http://www.adafruit.com/products/870

These displays use I2C to communicate, 2 pins are required to interface. There are multiple selectable I2C addresses. For backpacks with 2 Address Select pins: 0x70, 0x71, 0x72 or 0x73. For backpacks with 3 Address Select pins: 0x70 thru 0x77

Adafruit invests time and resources providing this open source code, please support Adafruit and open-source hardware by purchasing products from Adafruit!

Written by Limor Fried/Ladyada for Adafruit Industries.

BSD license, all text above must be included in any redistribution

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

#include <Wire.h>

#include "Adafruit\_LEDBackpack.h"

#include "Adafruit\_GFX.h"

Adafruit\_BicolorMatrix matrix = Adafruit\_BicolorMatrix();

void setup() {

Serial.begin(9600);

Serial.println("8x8 LED Matrix Test");

matrix.begin(0x70); // pass in the address

}

void loop() {

matrix.setTextWrap(false); // we don't want text to wrap so it scrolls nicely

matrix.setTextSize(1);

matrix.setTextColor(LED\_GREEN);

for (int8\_t x=7; x>=-36; x--) {

matrix.clear();

matrix.setCursor(x,0);

matrix.print("REThink");

matrix.writeDisplay();

delay(100);

}

matrix.setTextColor(LED\_RED);

for (int8\_t x=7; x>=-36; x--) {

matrix.clear();

matrix.setCursor(x,0);

matrix.print("Rocks");

matrix.writeDisplay();

delay(100);

}

matrix.setRotation(0);

}

# Appendix E

**Activity – Sample worksheet on microprocessor hardware**

**In groups, find photographs of the following microprocessor components and describe what they do:**

In your groups, define each of the following items and put a picture of it next to the description:

1. Breadboard
2. Capacitor
3. Power Regulator
4. Heatsink
5. Resistor
6. Integrated Circuit (IC) chip
7. Jump Wires
8. Microcontroller

# Appendix F

**Make your own 8-Bit Art!**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# References

Bell, T., Witten, I. H., & Fellows, M. (2006). CS Unplugged An enrichment and extension programme for primary-aged children. Retrieved from http://csunplugged.org/

Chapman, G. and Goode, J. (2013). Exploring Computer Science, version 5. Retrieved from http://www.exploringcs.org/curriculum

Naps, T., Rößling, G., Dann, W., Almstrum, V. , Fleicher, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., & Velazques-Inturbide, J. (2002). Exploring the role of visualization and engagement in computer science education. *ACM SIGCSE.* Retrieved from http://dl.acm.org/citation.cfm?id=782998

Salvucci, D. (2009, June). Rapid Prototyping and Evaluation of in-vehicle interfaces. *ACM Transactions on Computer-Human Interactions*, p. 33.