CS Makers – Traffic light with LEDs

For this project we will build on what we learned from the "Blink a LED" circuit we built previously. This project will be a traffic light that automatically changes when it detects a car. As before, we will need to build the circuit and then write a control program.

You will need three (3) LEDs, one red, one yellow, and one green LED. You will also need three (3) resistors. You need a resistor that is at least 150 Ω (Ohms) and no more than 330 Ω. Remember it is important to use resistors to make sure that we don't damage either the LEDs or the Micro:bit. The circuit diagram shown in Figure 1 illustrates Pin 0, Pin 1, and Pin 2 of the Micro:bit are connected to the three LEDs (one pin per LED), then to the resistors, then all three are connected to the ground (GND) pin on the Micro:bit.

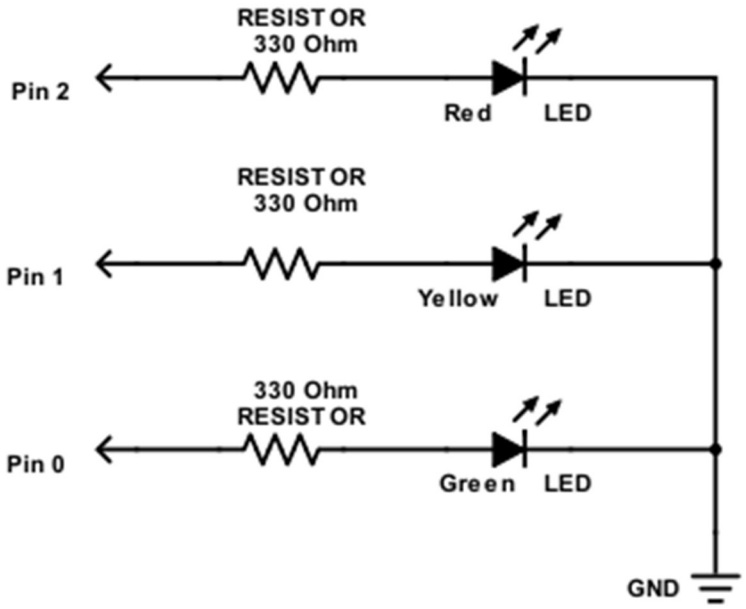

Figure 1. LED Circuit

Parts required:

- 1 Micro:bit
- 3 LEDs (red, yellow, green)
- 3 330 Ω resistor
- 1 Breadboard
- 4 Alligator (crocodile) clips
- 1 Battery pack and batteries –OR– USB cable for power

Remember that wires and resistors will conduct electricity in any direction. This means it does not matter which way you connect alligator clips (a.k.a. crocodile clips) or resistors. However, a LED will only conduct electricity in one direction. The schematic shown in Figure 1 tells us which way to connect the LED once you know which wire on the LED is marked positive (+) and which one is marked minus (-). Figure 2 shows us how to distinguish the wires on a LED. Notice the longer wire is the positive.

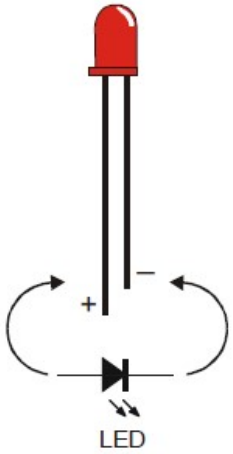

Figure 2. Light Emitting Diode

As we build this circuit, we need to learn about how our breadboard works. Figure 3 shows our breadboard with some areas highlighted with colors. In general, the purpose of a breadboard is to allow us to construct circuits temporarily. Makers, computer scientists, and engineers call this prototyping. It is called a breadboard because when people first prototyped circuits they used slabs of wood and a breadboard was often the handiest piece of wood around the house.

The holes in the breadboard are exactly the right size for LEDs, resistors, jumper wires, and other electronic parts. Some of the holes are connected to each other so that we can attach electronic components together without soldering, using electrical tape, or other fasteners. However, not all the holes are connected together and knowing which ones are connected and which ones are not is important so that our circuits will work properly.

The areas surrounded by orange rectangles in Figure 3 are called the power rails. All of the holes adjacent to each red line (positive power rail, marked +) are connected together on each side. The positive rail on the left side is not connected to positive rail on the right side. The same is true for the two rails next to the black lines (negative power rail, marked -). The power rails are often used to connect power and ground (GND) to the breadboard.

The interior of the breadboard is divided into two halves. The two halves of the interior of the breadboard are separated by a ravine or gutter shown by the green arrow in Figure 3. The two halves separated by the ravine on the

Formally, the ravine in the breadboard is called “DIP support”. The acronym DIP stands for *dual inline package* and describes the packaging for one kind of computer chip. The ravine allows us to use these chips with our breadboard.

breadboard are not connected together. For example, the hole at a8 is not connected to f8. We will often place certain parts such as switches and chips across the ravine to separate their wires.

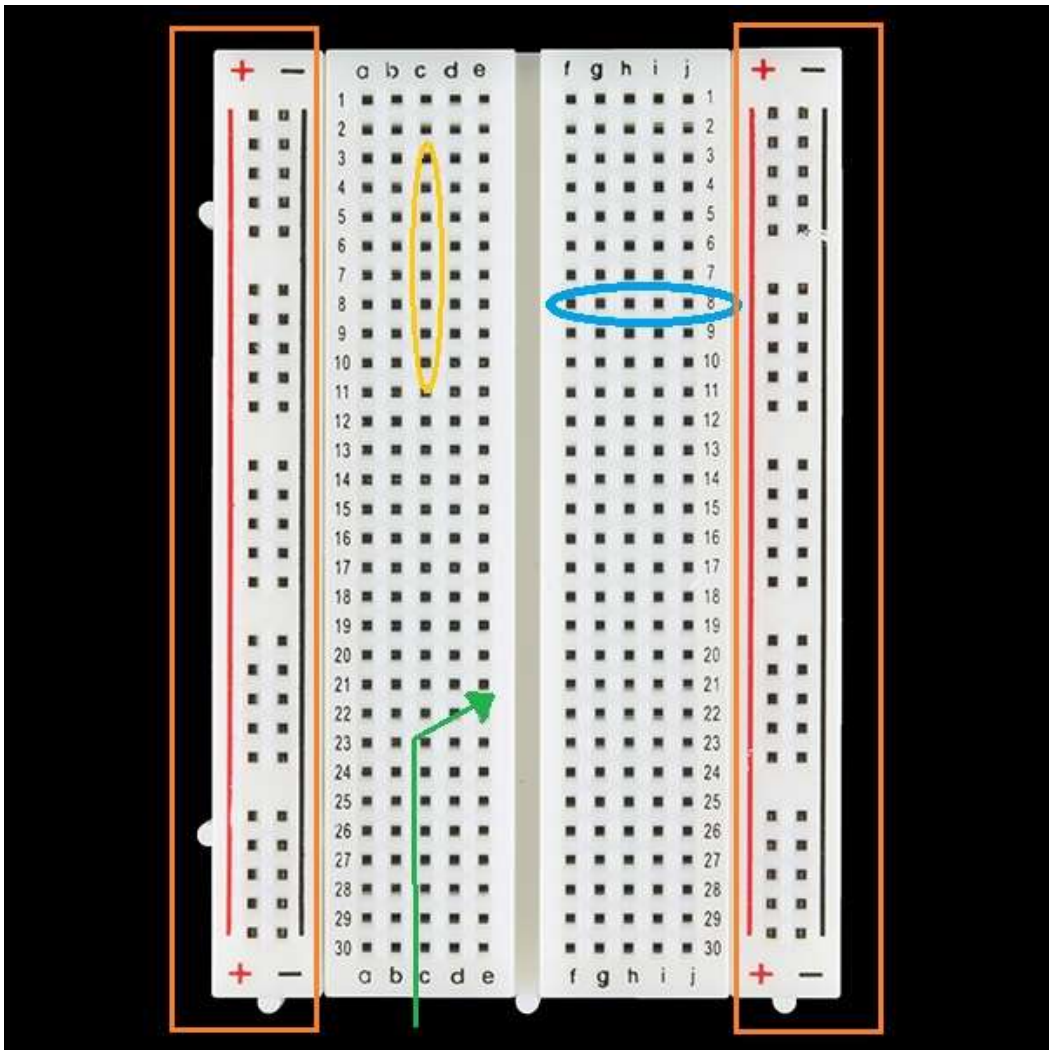

Figure 3. Breadboard for prototyping circuits

In Figure 3 the columns are marked with letters a-e and f-j and the rows are marked with numbers 1-30. The holes on each row either side of the ravine are connected. For example, f8 is connected to g8, h8, i8, and j8 shown in the blue oval on Figure 3. However, row number 8 on the left of the ravine (a8 – e8) is not connected to f8-j8. The holes in the columns are not connected. For example, the holes in the column c3 – c10 shown in the yellow oval in Figure 3 are not connected.

You can find out more about how breadboards work and how to use them to prototype circuits on the Sparkfun web site https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard.

Assemble the circuit on a breadboard as shown in Figure 4. You should use alligator clips to connect the blue wires to the Micro:bit. You must wire the circuit exactly as shown on the breadboard. Notice in Figure 4 that the resistors are placed across the ravine of the breadboard so that one end of the resistor is on either side (remember the two halves of the breadboard on either side of the ravine are NOT connected). Figure 4 shows which rows and columns are used on the breadboard with faint green shading. In this image, the negative power rail is blue and is used to connect the negative side of the LEDs to GND. The positive wire on the LED is connected to the resistor (see Figure 1).

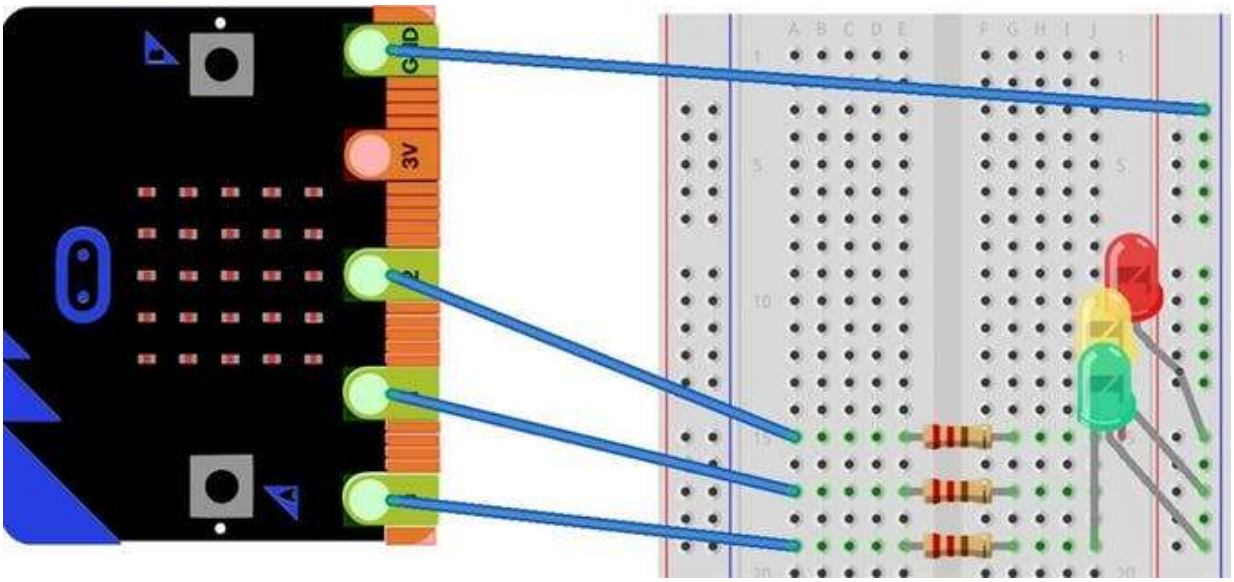

Figure 4. Traffic light LED circuit

Now we are ready to design our program to control our traffic light LEDs. The program is similar to our previous project where we blinked a LED on and off. Our traffic light start with a red light and wait for a “car” to arrive. When a vehicle arrives our “car detector” will let our program know that a car has arrived. We will use a pushbutton to detect when a car arrives. Because computers can do things really fast and making the computer check hundreds of times per second for a car wastes energy, we will only check our “car detector” once per second. Once a car arrives, we will change the light to green and start a timer. The timer will determine how long the light is green and will count 3 seconds. After the 3 second timer expires, the light will turn yellow and we’ll start a second timer to count to 2 seconds. After the 2 second timer expires we change the light to red and wait for the next car to arrive. Figure 5 shows our algorithm as a flowchart.

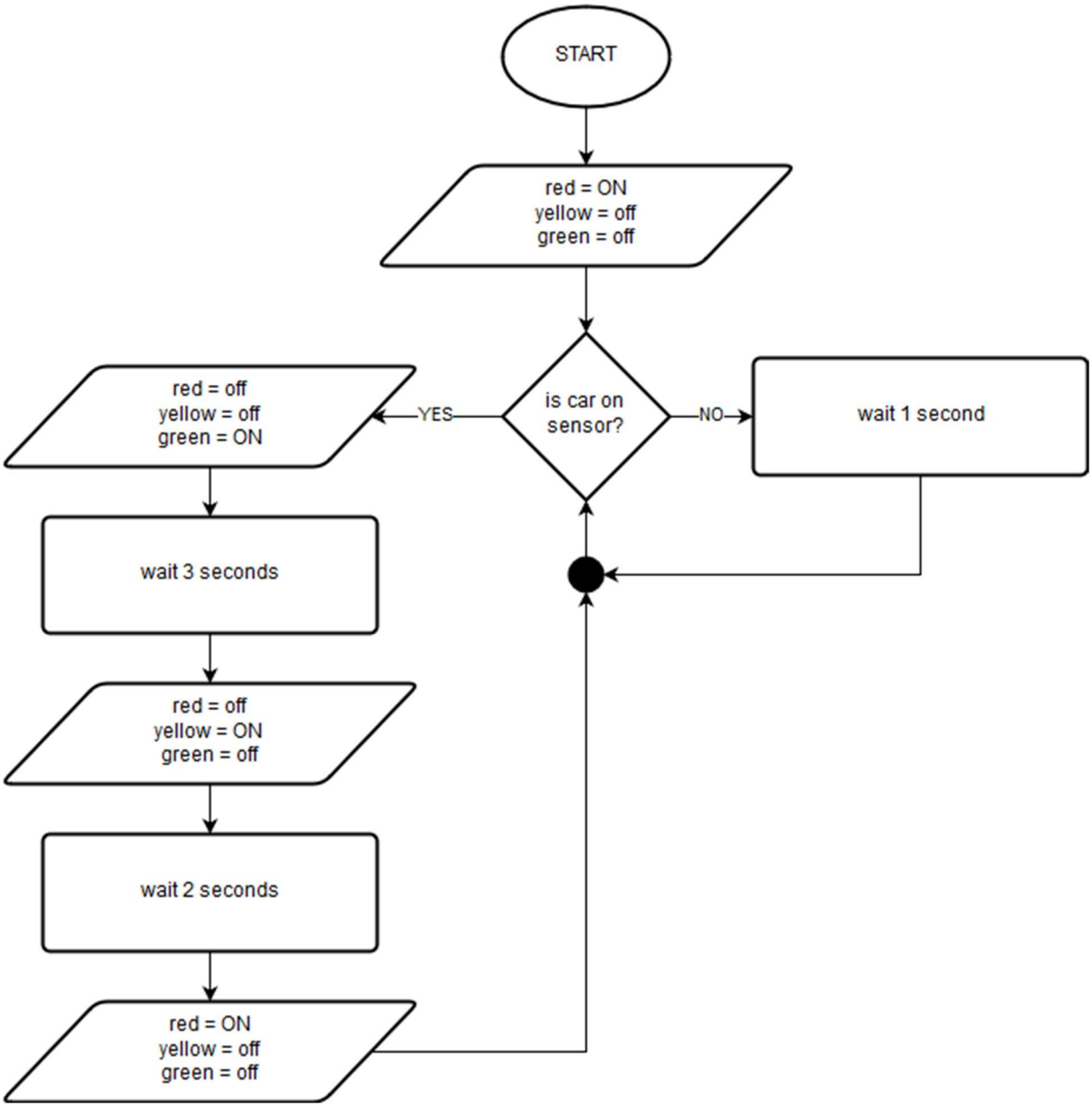

Figure 5. Traffic light algorithm

Open up the Javascript Blocks Editor for the Micro:bit and create your program. Recall that most microcontroller programs have two sections, a start (or initialization) section and a forever section (the part that repeats forever). We need to initialize the LEDs in the **start** block so that the red LED is on and the yellow and green LEDs are off. The remainder of our Micro:bit program will be inside the **forever** block.

Recall that on and off (which is the same as high and low) is often denoted by the numbers one and zero in computer science. On or high is denoted as the number one and off or low is denoted as the number zero.

0 (zero) = Off

1 (one) = On

We will use the same block that we used to toggle the single Pin in our “Blink a LED” program to toggle (turn on and off) all three Pins that control our LEDs in this program. Access the “Pins” drawer under “Advanced” to find the digital write pin block (see Figure 6).

Figure 6. Digital write block in the Pins drawer

In addition to the “digital write pin” block you will also need the “pause” block and the block to detect when Button “A” is pressed (look in the Input drawer for the pushbutton block).

Create your program using the flowchart shown in Figure 5 then follow the same steps your learned in the “Introduction to the Micro:bit” lesson to load your program into your microcontroller. If your program does not work as expected, carefully review your circuit to make sure it matches Figure 4 and verify that your program matches Figure 5. The complete program is shown on the next page in Figure 7.

Figure 7. Blink a LED program

Computer Science topics – Algorithms, flowcharts

Figure 2 - https://learn.parallax.com/tutorials/robot/shield-bot/robotics-board-education-shield-arduino/chapter-2-shield-lights-servo-17

Figure 6 - https://microbit.hackster.io/anish78/traffic-light-system-using-bbc-micro-bit-da2f47

All other figures and text James A. Jerkins, Ph.D., CISSP