TOBB
UNIVERSITY OF
ECONOMICS AND TECHNOLOGY

Programming Manual for Sparki

ROB-ETÜ

MoRS

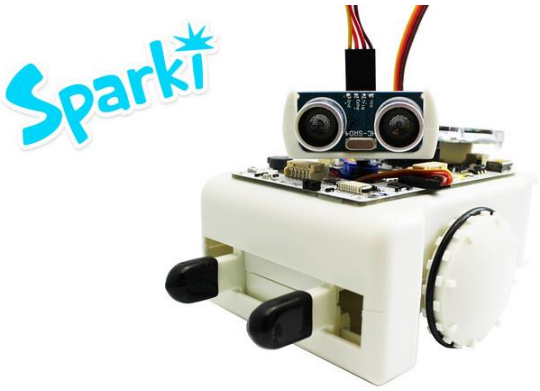
Mobile Robotic Systems Laboratory

2015

Contents

Sparki – The Easy Robot for Everyone!

Sparki is an affordable, easy to use, and fun intro to programming, electronics, and robotics. It is an open-source, Arduino robot designed for students, late-elementary age and above that comes pre-assembled and packed with 12+ sensors and features like remote control and drawing.



It has two programming environments: SparkiDuino, based on Arduino; and Minibloq, a drag-and-drop program like Scratch. It is great for families, STEM/robotics/programming educators, and DIY enthusiasts looking for a feature-packed platform.

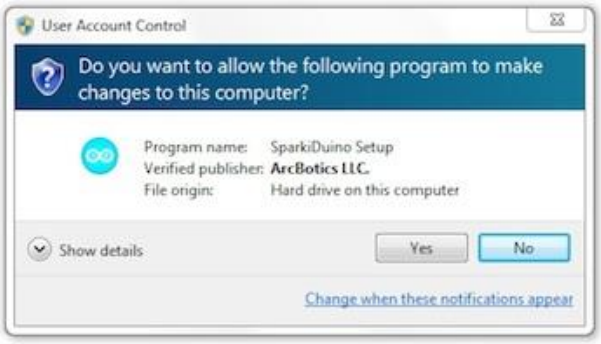
Sparki works out of the box with its remote control. To write your own programs, just plug it in via USB, install the custom-enhanced Arduino software and try any of the dozens of example programs. Possible activities to do with every sensor and actuator on Sparki:

- Detect distance to walls and objects with Ultrasonic HC-SR04 distance sensor
- Detect pick-ups, falls and climb hills with 3-Axis accelerometer
- Sense magnetic fields and detect compass heading with 3-Axis magnetometer
- Follow light and darkness with Light-sensing phototransistors
- Do mazes, follow lines, and sumo with Line-following and edge detection IR sensors
- Display data and graphics with 128×64 Graphic LCD
- Generate any color with an RGB LED
- Create musical tones and beeping with Buzzer
- Message other Sparkis with IR transmitter
- Control via remote control with IR receiver
- Control Sparki with lots of buttons with IR remote control
- Talk to an Arduino/Raspberry Pi with TTL serial port for expansion
- Talk with phones and computers with Bluetooth Serial Module Port
- Move precisely with measured movement down to millimeters/sub-degrees with stepper motors
- Programs easily with USB cable
- Powered by 4xAA batteries (rechargeable or alkaline, batteries not included)
- Draw and write precisely with a Sparki-fitted black marker

Installing SparkiDuino on Windows

1. Download the SparkiDuino software installer from this link[1].
2. Double click the installer icon:

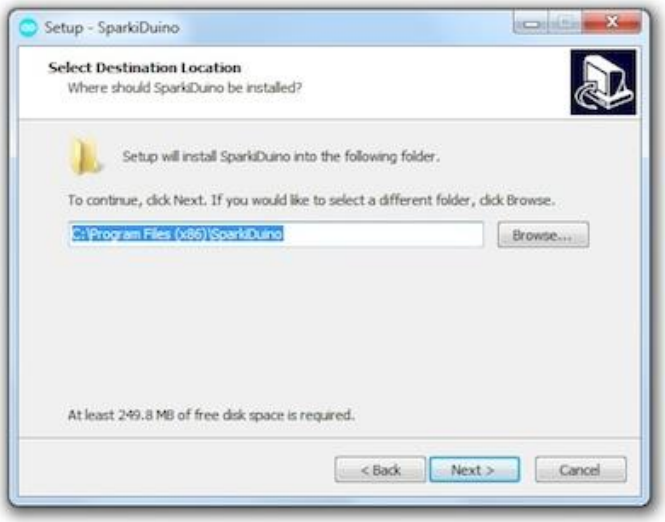
3. Click “Yes” to install:



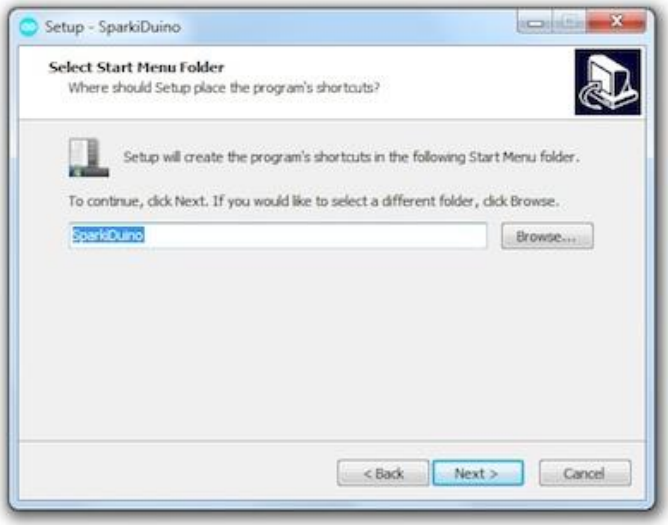
4. Then click “Next”:

[1] http://download.arcbotics.com/SparkiDuino-1.0.6.1-Setup.exe

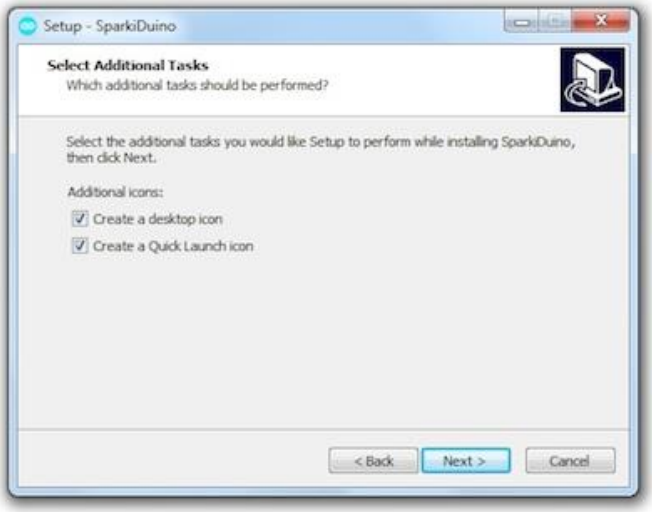
5. Choose where you want to install it:



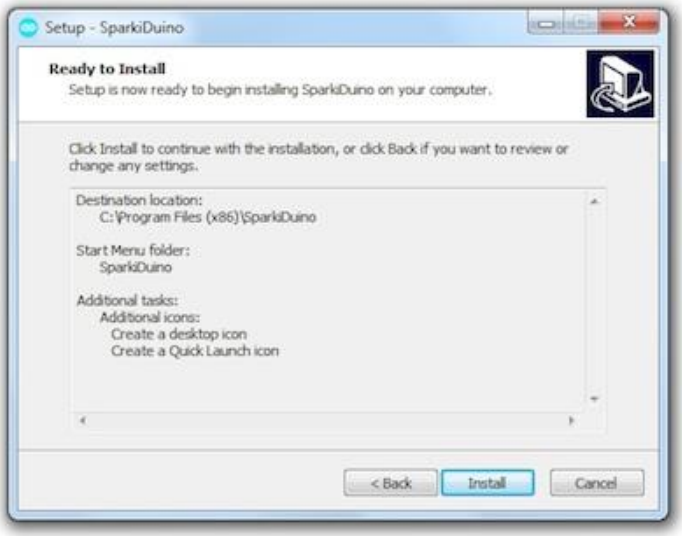
6. Choose what name you want to show up in the start menu:



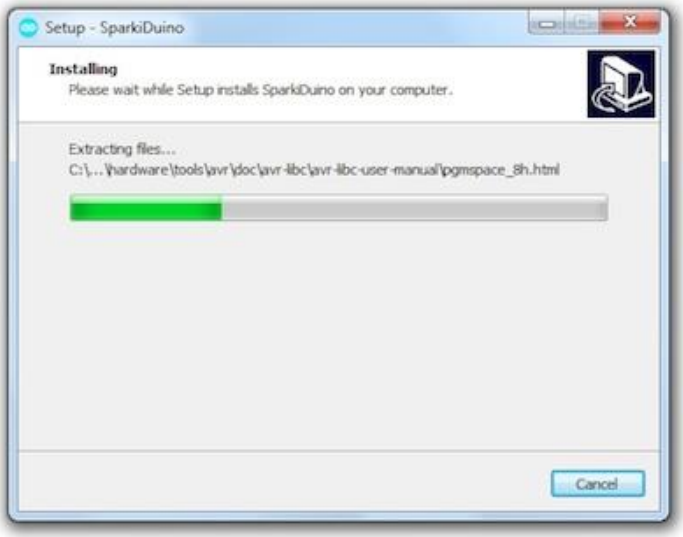
7. Choose what icons you want, and where you want them:



8. Confirm your choices and click “Install”;



9. Wait for the install to complete:



10. Now install the drivers by clicking “Next”:

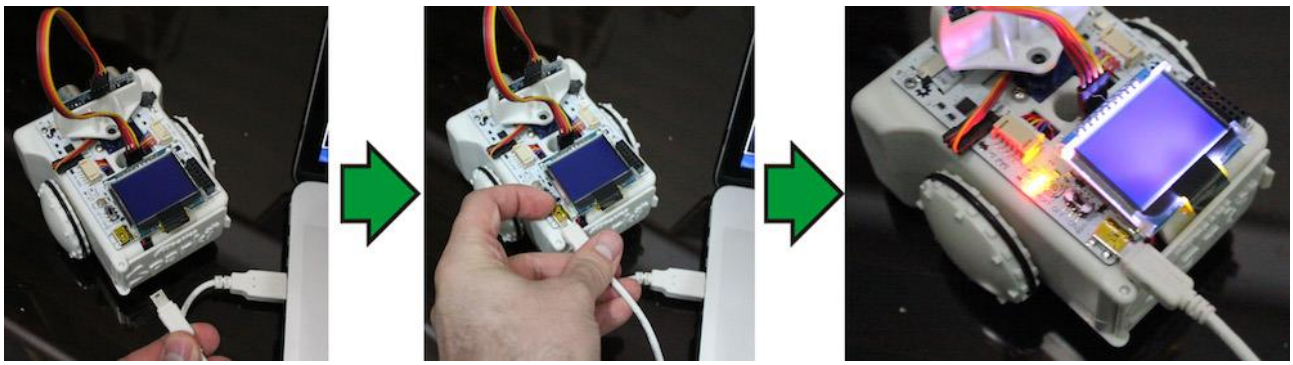
11. Once they're installed, click "Finish":

12. SparkiDuino is installed! Click "Finish" to launch SparkiDuino:

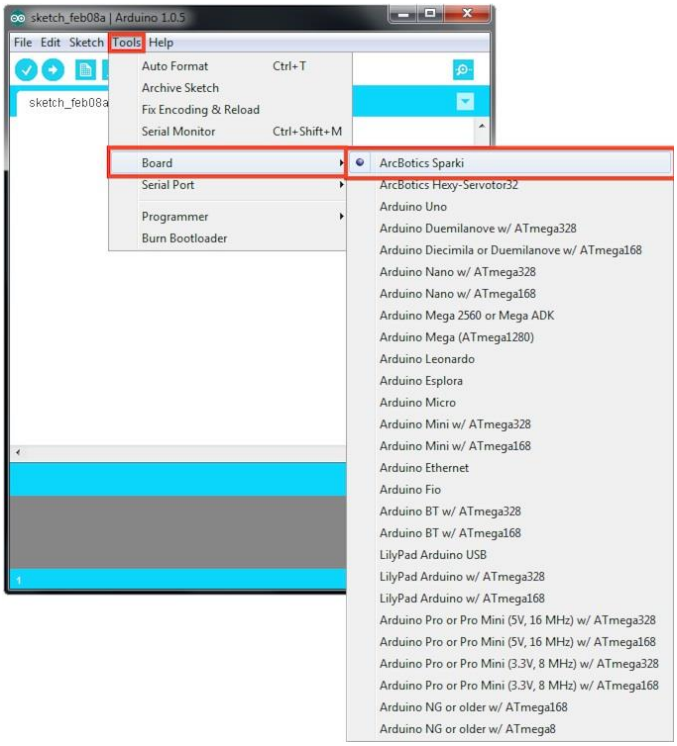
13. The SparkiDuino software should start automatically.

Uploading Code to Sparki

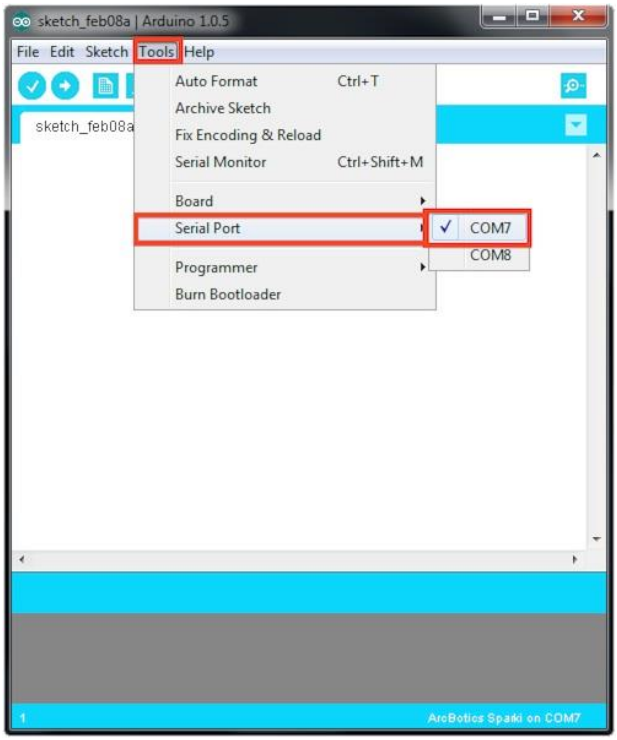
With the SparkiDuino software installed, plug in your Sparki's USB cable into your computer's USB port and Sparki. You cannot program Sparki with the bluetooth module.



Select the ArcBotics Sparki board by selecting it from Tools | Board.

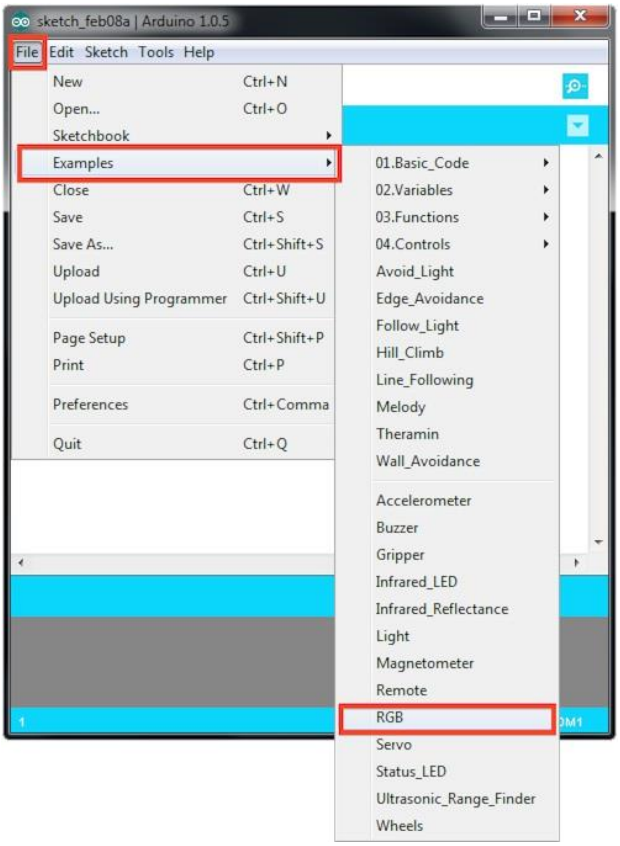


Select the serial device of the Arduino board from the Tools | Serial Port menu.



To have some code to upload, we will use one of the examples – the RGB – from the Sparki examples. All example code and code you write upload the same way.

Here we are selecting the RGB example. When this RGB code is selected and uploaded, the RGB LED on Sparki will rotate between Red, Green and Blue.



With this code selected, it should show up in the code box now. You can press the second button from the left (the left-pointing arrow). The code will first compile, then upload:

While the code is being uploaded, Sparki's status light first pulses red while in programming mode, then blinks rapidly while the program is being sent. It then goes solid red, off, then runs the program. And show it is done uploading:

Manually Uploading Code

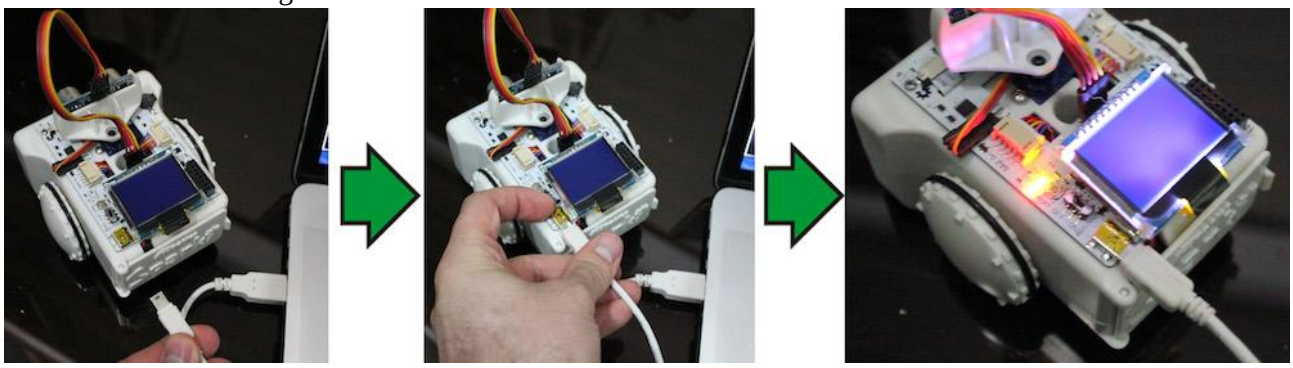
Attention!

This part is necessary if only you have a problem while loading code. Otherwise, please ignore this part.

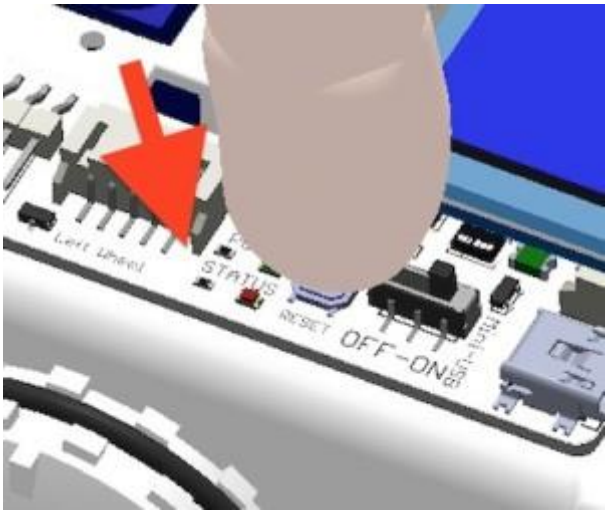
For proper installation and not causing any damage to the robot or your computer, please first read and understand these instructions, and then use them.

If Sparki ever gets stuck in a program, or for some reason the code isn't uploading you can always do a manual program upload.

1. Get everything ready:
    - Sparki's USB cable is plugged both into Sparki and the computer running SparkiDuino.
    - SparkiDuino and drivers are installed.
    - SparkiDuino is open and has a program ready to install.
    - If you are expecting Sparki to move, make sure the batteries are installed and the power switch is turned on.
    - Make sure the board and serial port are selected. If there is no serial port listed, this method still might work.

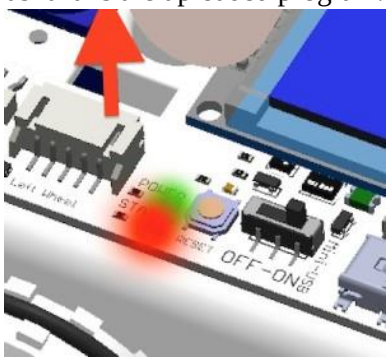


2. Hold down the Sparki's Reset button:
   The button is located next to the power switch and USB port on Sparki, to the left of the LCD display on Sparki's back.



3. Press Upload, wait for “Uploading…”:
   Press the “Upload” button in SparkiDuino. SparkiDuino will first say “Compiling Sketch” in the lower left-hand corner, then “Uploading…”.

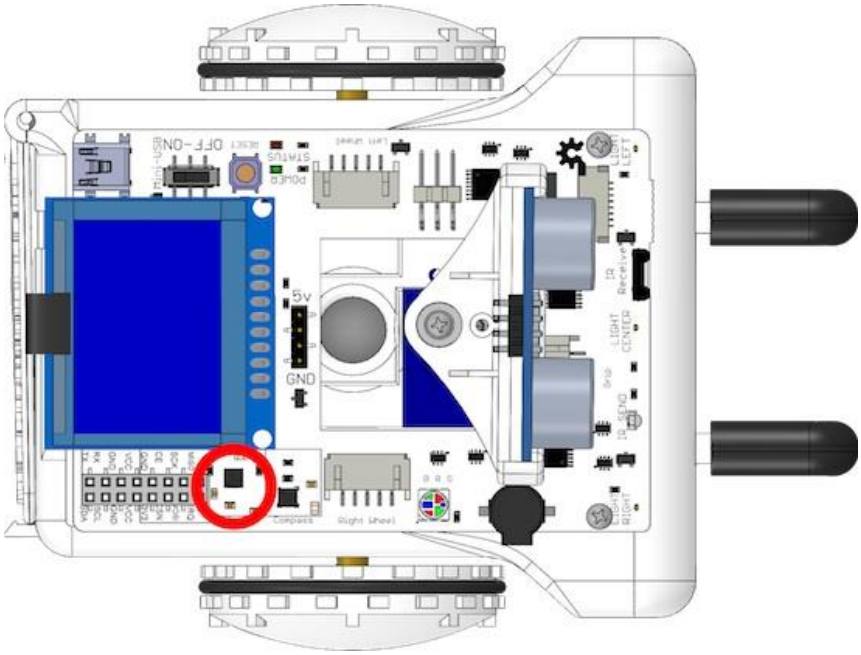
4. Release the Reset button:
   As soon as SparkiDuino says “Uploading…,” release the button. Sparki will boot into programming mode, and SparkiDuino will connect and upload the code. Sparki’s red led will blink rapidly while the code is being uploaded, then clear as it runs the uploaded program.



5. Done.

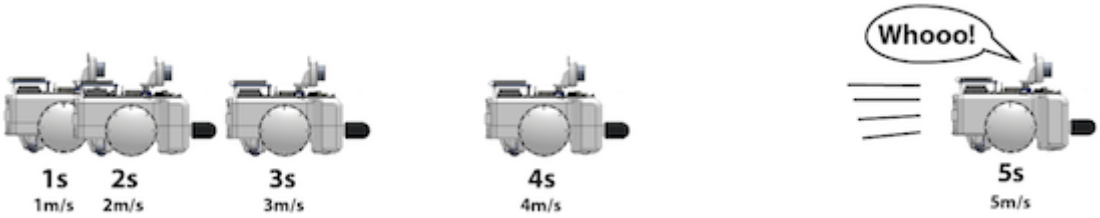
Accelerometer

Sparki has a 3-Axis accelerometer. It is used to detect the acceleration Sparki is experiencing. Usually, this mostly means the force of gravity. It can do this in all 3 XYZ axis, or directions, which are left/right (X), forward/backwards (Y), and up/down (Z). This is the same part that smartphones use to tell how you're tilting them.

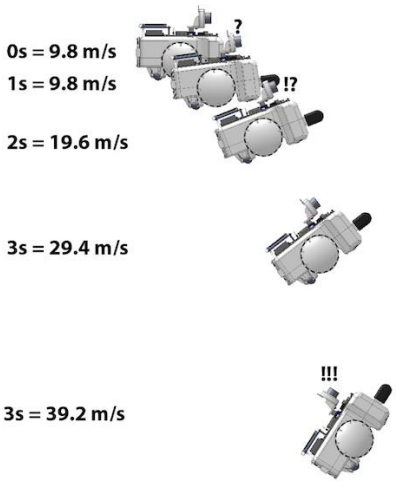


The Acceleration

Acceleration is how fast something speeds up. If we took a super turbo-charged Sparki and had it going 10 meters per second (36kph, 22 mph), the velocity is 10 meters per second, written as 10 m/s. If the car is reaching 10 meters per second in 10 seconds, you can say the acceleration is 1 meters per second per second, or 1 meters per second squared, written as 1 m/s^2.



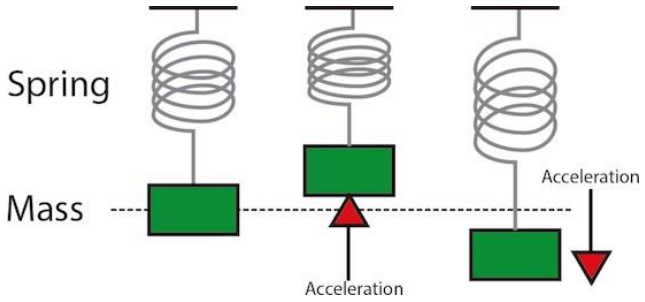
Gravity is also a force that causes acceleration. Objects that are falling will fall faster and faster with acceleration.



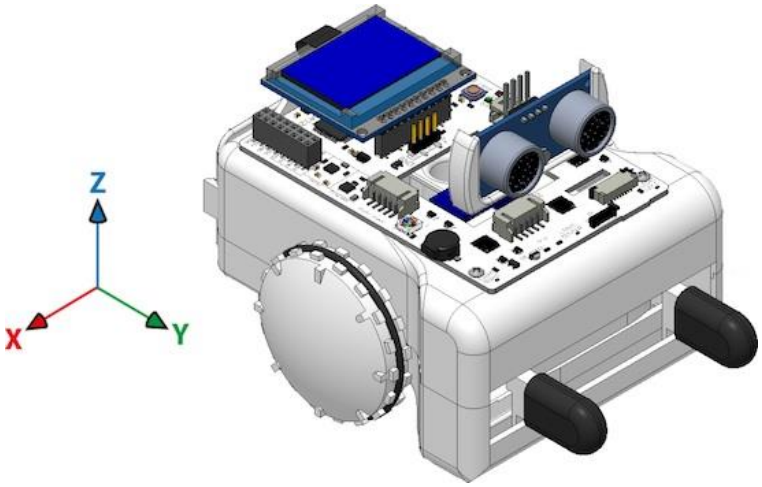
The only thing that keeps most things from falling is that they are being pushed up by the ground. Gravity is the acceleration that Sparki measures when it is sitting flat on a surface.

How it works?

The accelerometer is what's called a microelectromechanical, or MEMs device. MEMs are made of very tiny mechanical parts and wires too small to see, made in the same way microchips are made. Inside of the part, too tiny to see, are tiny springs. When the part is accelerated, these springs compress or relax. By measuring the amount these springs compress or relax, the part is able to measure the acceleration it is experiencing.



The part has 3 different axis, or directions, of these spring-masses. They are left/right (X), forward/backwards (Y), and up/down (Z).



The direction of the arrows shows the positive direction of the reading. If the acceleration is in the opposite direction, the reading will be negative. For example, if Sparki is sitting flat on a level table, the Z axis reading should be -9.8 m/s^2. If Sparki is then flipped upside-down, the reading would be positive 9.8 m/s^2.

Using the part

With the basic Sparki code in place, you can measure the acceleration in each axis using these commands:

```
sparki.accelX();
sparki.accelY();
sparki.accelZ();
```

This command returns the acceleration the accelerometer reads in meters per second squared (m/s^2), in the form of a float. For example, if Sparki is sitting flat on a level table, `sparki.accelZ();` should read -9.8 m/s^2. For conversion sake, 1G = 32 ft/s^2 = 9.8 m/s^2. This is also known as 1G, or one unit of Earth's gravity.

SparkiDuino already has code examples for you to use:

File > Examples > Accelerometer

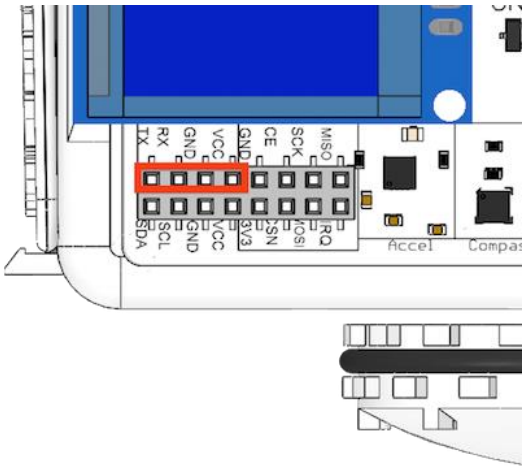
```
/*****************************************
 Basic Accelerometer Sensor test

 Sparki has a 3-Axis accelerometer. It is
 used to detect the acceleration Sparki is
 experiencing. Usually, this mostly means the
 force of gravity. It can do this in all
 3 XYZ axis, which are left/right (X),
 forward/backwards (Y), and up/down (Z). This
 is the same part that smartphones use to tell
 how you're tilting them.

 This program shows how to read the sensor
 and display the information on the LCD.
*****************************************/
#include <Sparki.h> // include the sparki library

void setup()
{
}

void loop()
{
  sparki.clearLCD(); // wipe the screen

  float x  = sparki.accelX();   // measure the accelerometer x-axis
  float y  = sparki.accelY();   // measure the accelerometer y-axis
  float z  = sparki.accelZ();   // measure the accelerometer z-axis

  // write the measurements to the screen
  sparki.print("Accel X: ");
  sparki.println(x);

  sparki.print("Accel Y: ");
  sparki.println(y);

  sparki.print("Accel Z: ");
  sparki.println(z);

  sparki.updateLCD(); // display all of the information written to the screen
  delay(100);
}
```

Bluetooth Module

Some Sparkis come with a Bluetooth Module. It is used to communicate wirelessly with your Sparki. While you cannot use it to program Sparki, it is useful for communicating with Sparki via serial commands wirelessly.

Using the part

The Bluetooth Module is inserted into Sparki's expansion port pins. When plugged in, the module should light up and blink. The pins you will plug the Bluetooth Module into on Sparki are named TX, RX, GND and VCC. The Bluetooth Module may be a tight fit next to the LCD, that's ok, as long as the Bluetooth Module is plugged into the correct pins it will function properly, even if it's pushed up against the LCD screen.



Here's a quick explanation of the pins on Sparki that you will plug the Bluetooth Module into:

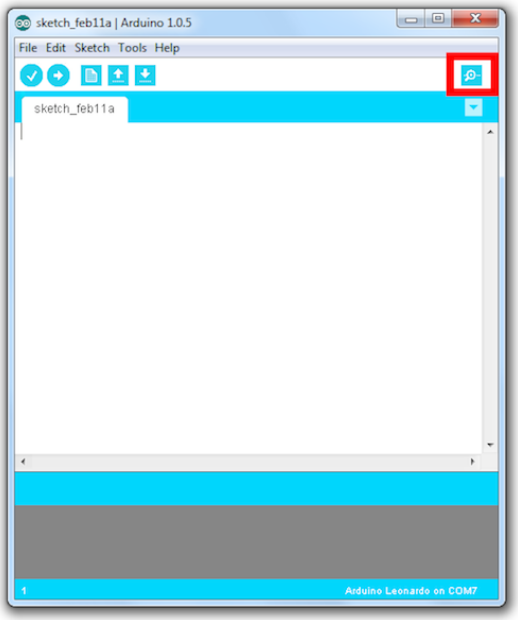
- **TX** – This is the pin that Sparki uses to transmit information out of Sparki and to the Bluetooth for transmitting. Because the Bluetooth is receiving information from Sparki on Sparki's TX pin the RX Bluetooth pin plugs into this pin.
- **RX** – This is the pin that Sparki uses to receive information from the Bluetooth. For that reason the TX Bluetooth pin plugs into this pin.
- **GND** – This supplies a ground path for the electricity to travel back from the Bluetooth module to Sparki.
- **VCC** – This supplies the source of electricity for the Bluetooth Module from Sparki.

Once the module is blinking, you can connect it to your computer. You'll need to pair your Bluetooth Module at least once every time you use Sparki with a new computer or device, but after that the computer or device will probably remember about Sparki's Bluetooth Module unless something happens to your computer or device Bluetooth preferences or files.

After you connect your Bluetooth Module it might go from displaying on your computer or device as a bunch of numbers and some dashes to being called "ArcBotics."

Now, with the module connected with your computer, you can access it as a serial port. To find the serial port select Tools, go down to Serial Port and look at the options you have to select. There should be one or two new serial ports that you can select. The serial port is most likely called something like "ArcBotics-DevB."

You can access the serial terminal in Arduino by pressing the upper right-hand corner in the Arduino software.



The serial terminal will allow you to send information to Sparki and see anything that Sparki is broadcasting over the Bluetooth Module. If Sparki has commands that tell it what to do with the information you send it, you can even control Sparki's output with this serial monitor.

Previously you've used serial communication to upload code onto Sparki over the USB serial port. Now that you've moved on to exploring the Bluetooth Module you'll be getting comfortable with serial ports since the Bluetooth Module also uses a serial port.

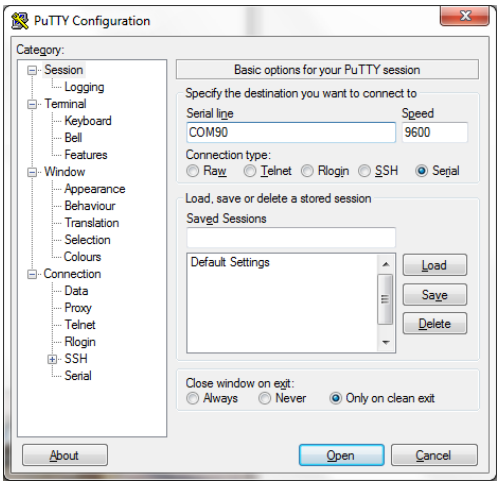
Sparki has two serial ports:

1. The USB serial port, which is accessed in the code you load onto Sparki via Serial.
2. The port the bluetooth module plugs into is usable via Serial1 in the code you will load onto Sparki. To use the bluetooth module to communicate instead of the USB port, change any commands that use Serial to use Serial1 instead.

The first test that we should do to ensure that our Bluetooth module is working fine is to run the *Hello World* program, but where we used *Serial* we want to change that to Serial1. Also add the `Serial1.begin(9600)` command to the setup function. This command tells Sparki that the communication we'll be sending over the Bluetooth Module will be at a speed of 9,600 bits per second or a baud rate of 9600. It's important to be aware of this number because it needs to match in the code running on Sparki and in the program on your computer or device you use to receive the communication from Sparki. If Sparki is "speaking" at a faster or slower speed than your receiving device can "listen" (if the baud rates don't match up) then the communication will be show up as gibberish.

```
/*****************************************
 Printing text using the Serial (Bluetooth) port

 http://arcbotics.com/lessons/using-usb-serial-and-bluetooth-communications/

*****************************************/
#include <Sparki.h>  // include the sparki library

void setup()
{
  Serial1.begin(9600);
}

void loop()
{
  Serial1.println("Hello World");
  delay(1000);
}
```

On computers running Windows, using the Bluetooth communications will require a bit of extra effort, since the Arduino terminal (which is the terminal used in SparkiDuino's software) may not be able to work with Bluetooth serial ports. So, to ensure that everything works properly, we will install a better terminal program. There are a lot of free terminal programs out there, but one of the best is PuTTY[2]. This flexible and reliable software will allow us to work easily with any serial connection, being it over Bluetooth, USB, etc… Once you have downloaded PuTTY, you just need to run it. In the first screen, please select Serial and configure the port of your Bluetooth connection, as well as the Speed, which should be 9600.



Then press the Open button. Your Bluetooth module in the Sparki robot should stop blink (the LED there will be ON permanently while a connection is established), and the following screen will appear.

[2] http://www.putty.org/

Let's test bidirectional communications, but instead of just running the echo program for Serial1, we will make something a bit more interesting: a bridge between the USB and the Bluetooth ports. With it, we will be able to send text from one port and see it on the other one. To do this though, we will need two terminals.

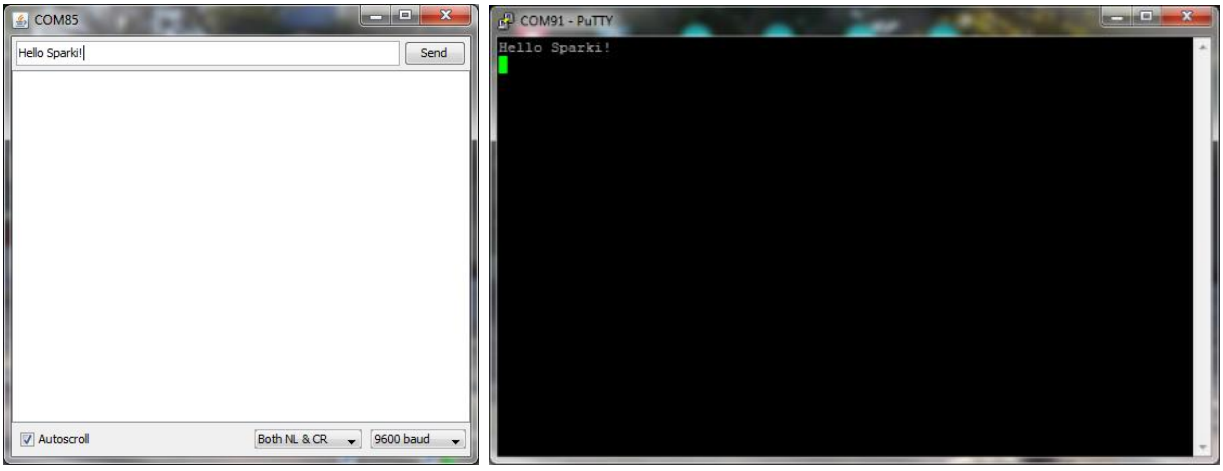
We can run two instances of PuTTY, or open the SparkiDuino's terminal for the USB port, and the PuTTY terminal for the Bluetooth.

Before we look at any terminals, here is the code, which is similar to the echo program, but a bit more complex since it deals with two ports instead of one:

```
/*****************************************
 Simple echo program using the Serial (Bluetooth) port

 http://arcbotics.com/lessons/using-usb-serial-and-bluetooth-communications/

*****************************************/
#include <Sparki.h>  // include the sparki library

void setup()
{
  Serial1.begin(9600);
}

void loop()
{
  if (Serial1.available())
  {
    int inByte = Serial1.read();
    Serial.print((char)inByte);
  }
  if (Serial.available())
  {
    int inByte = Serial.read();
    Serial1.print((char)inByte);
  }
}
```

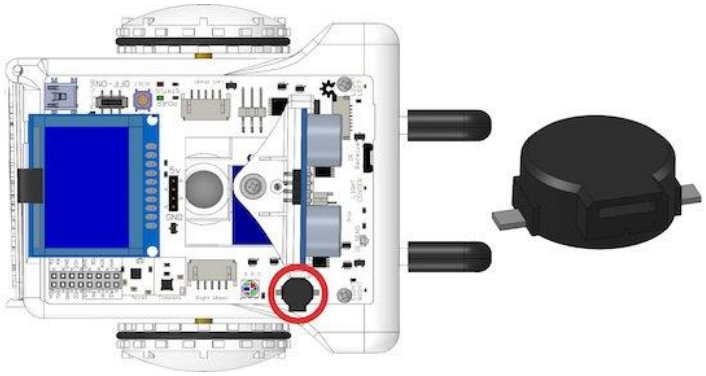
Please note that the received characters on Serial are sent to the Serial1, and vice versa.

Now that you have uploaded this program, you can open the terminals and type on one of them to see the text on the other.



Buzzer

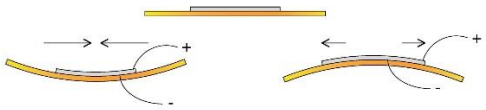
Sparki has a buzzer that can make sounds.



How it works?

The buzzer on Sparki is a piezo electric buzzer. Piezo electric materials change their shape when electricity is applied to them.

By turning voltage on and off to the piezoelectric material in the buzzer, the plate bends and moves the air back and forth.



This plate moving air back and forth compresses the air to make sound waves that you can hear.

Using the part

With the basic Sparki code in place, you can make Sparki beep using this command:

```
sparki.beep();
```

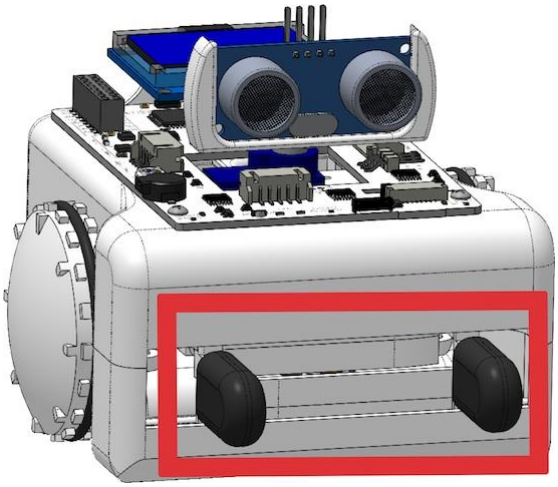
SparkiDuino already has code examples for you to use:

File > Examples > Beep

```
/*****************************************
 Basic Buzzer test

 Sparki has a buzzer on its back. Try
 making it beep!
*****************************************/
#include <Sparki.h>  // include the robot library

void setup()
{
}

void loop()
{
    sparki.beep(); // Sparki beeps!
    delay(1000); // wait a second (1000 milliseconds)
}
```

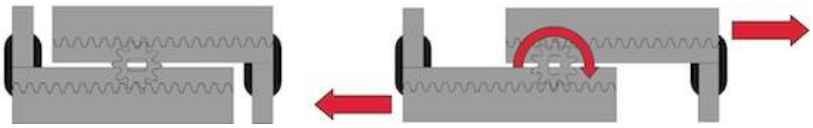
Gripper

Sparki has two little grippers that can grab objects and drag them around.



How it works?

Sparki's grippers are driven by the same stepper motor that the wheels use. The grippers are driven by rack and pinion style gear. For Sparki, each arm is a rack, and the stepper motor has the gear.



Using the part

With the basic Sparki code in place, you can move the gripper using these commands:

```
sparki.gripperOpen();
sparki.gripperOpen(width_cm);
sparki.gripperClose();
sparki.gripperClose(width_cm);
sparki.gripperStop();
```

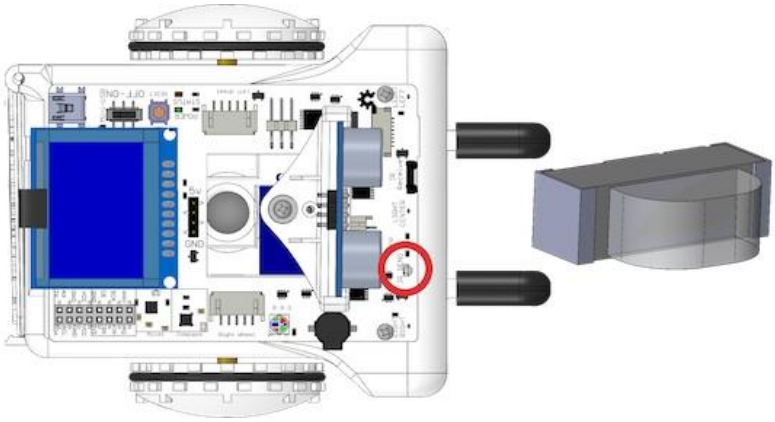
SparkiDuino already has code examples for you to use:

File->Examples->Gripper

```
/*****************************************
 Basic Gripper test

 Sparki has two little grippers it can be
 used to grab objects and drag them around.
 See what you can grab with them!

 http://arcbotics.com/products/sparki/parts/gripper/
*****************************************/
#include <Sparki.h> // include the sparki library

void setup()
{
}

void loop()
{
  sparki.gripperOpen();  // open the robot's gripper
  delay(1000);           // for 1 second (1000 milliseconds)

  sparki.gripperClose(); // close the robot's gripper
  delay(1000);           // for 1 second (1000 milliseconds)

  sparki.gripperStop();  // stop the grippers from moving
  delay(1000);           // for 1 second (1000 milliseconds)

  sparki.gripperOpen(3);  // open the robot's gripper by 3 centimeters
  sparki.gripperClose(3); // close the robot's gripper by 3 centimeters
  delay(1000);           // wait 1 second (1000 milliseconds)
}
```

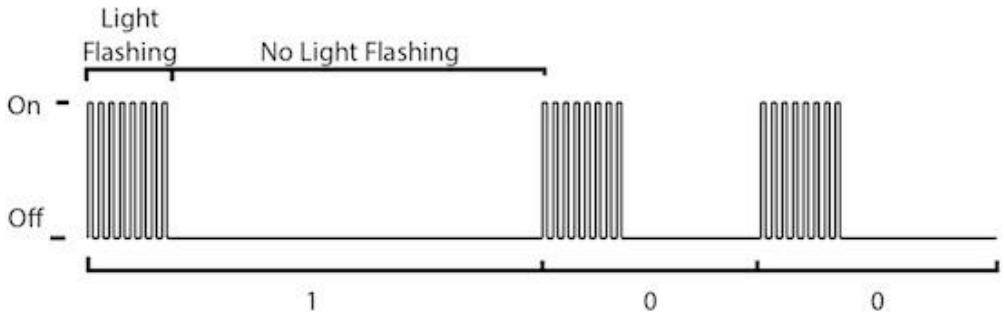
Infrared LED

Sparki has an Infrared LED. It is used to send information to other Sparkis like a remote control by rapidly blinking invisible infrared light.



How it works?

Infrared remotes work by sending codes via flashing lights. These lights flash really fast; Sparki's infrared LED flashes 38,000 times a second. By changing the amount of time between each flash, Sparki can send a burst that is meant to be a 1 or a 0, known as bits.



Sparki sends these flashes until an entire code has been sent. Specifically, Sparki uses the NEC infrared code protocol[3].

Using the part

With the basic Sparki code in place, you can send infrared codes using this command:

```
sparki.sendIR(code);
```

Where code is an 8bit value. This can be a number from 0 to 255, or a character like 'a' or 'D'. Sending a code like this will allow a Sparki nearby to receive it with the Infrared Remote Receiver.

[3] http://techdocs.altium.com/search/wikinode/NEC+Infrared+Transmission+Protocol%20type:wikipage

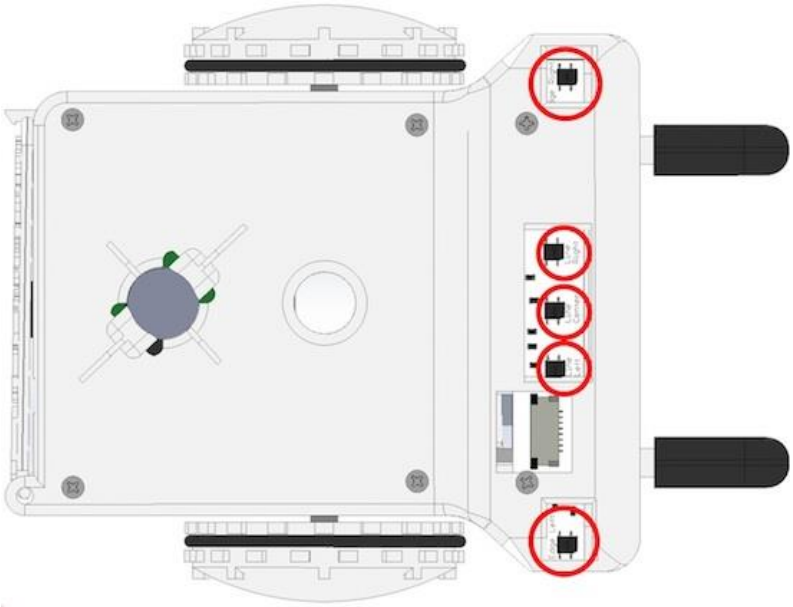
SparkiDuino already has code examples for you to use:

File > Examples > Infrared LED

```
/*****************************************
 Infrared LED Send Code example

 Sparki has an infrared LED. You can use it
 to communicate with other Sparkis and other
 devices that use infrared remote controls.
 Other Sparkis can receive the code using
 sparki.readIR();

 This example has Sparki sending a code every
 second. The code increases each time by 1.
 When the code reaches 255, the biggest number
 it can send, it goes back to zero and starts
 all over again.
*****************************************/
#include <Sparki.h> // include the sparki library

void setup()
{
}

int code = 0;
void loop()
{
  sparki.sendIR(code);
  code = code + 1; // increase code by 1

  // set code back to 0 if it goes over 255 - the highest number sendIR can send
  if(code = 255){
    code = 0;
  }

  delay(1000); // wait one second
}
```

Infrared Reflectance Sensor

Sparki has 5 infrared reflectance sensors underneath it.

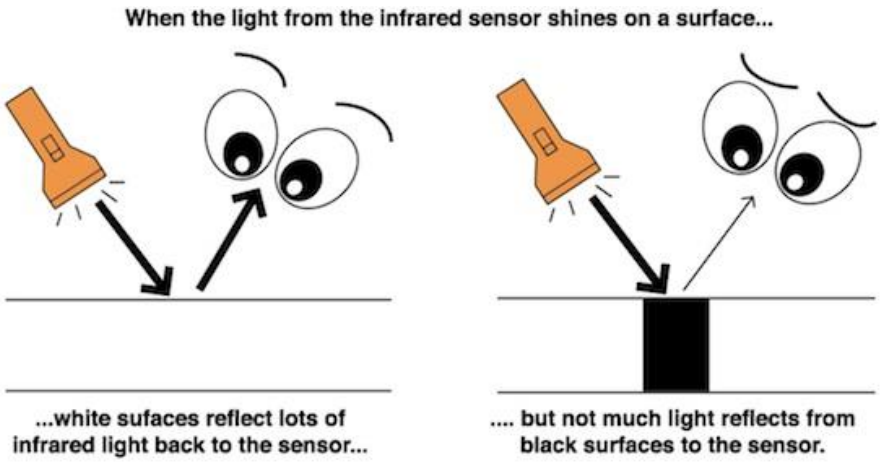


How it works?

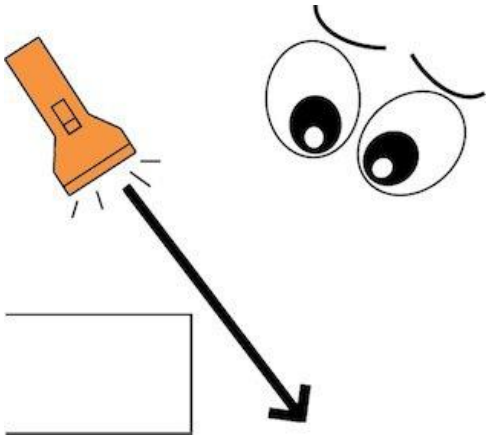
Each of these sensors has a phototransistor and an LED (Light Emitting Diode). These act like a pair of eyes and a little flashlight.

The LED is a little flashlight that shines invisible infrared light below the sensor.

The phototransistor is like a little eye that looks for how much infrared light comes back from the LED. The more light the phototransistor receives, the higher the voltage Sparki will read from it. You can use this to sense what kind of surface is underneath Sparki.



You can use this same idea to tell if Sparki is about to go over the edge of a surface, like a table. Without any surface underneath to bounce back the light, the sensor will not see much light.

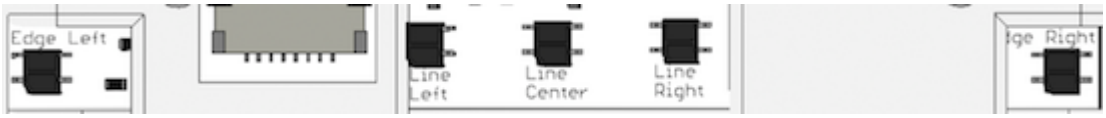


Using the part

With the basic Sparki code in place, you can measure an infrared reflectance sensor by using one of these commands:

```
sparki.edgeLeft();
sparki.lineLeft();
sparki.lineCenter();
sparki.lineCenter();
sparki.edgeRight();
```

Each of these commands matches the sensor names underneath Sparki, next to each sensor.



If you want to read the sensor marked Line Center for example, use the robot.lineCenter(); command.

These commands return the number the sensor reads in the form of an integer. A typical reading for bright white paper is around 1000. A typical reading for a black line is below 400. A typical reading for going off the edge of a surface is below 200. These numbers can change based on outside factors though, like how much sunlight (infrared light) a room has, how dark/white something is, how far away the sensor is from surface and more. Experiment and see what changes it!

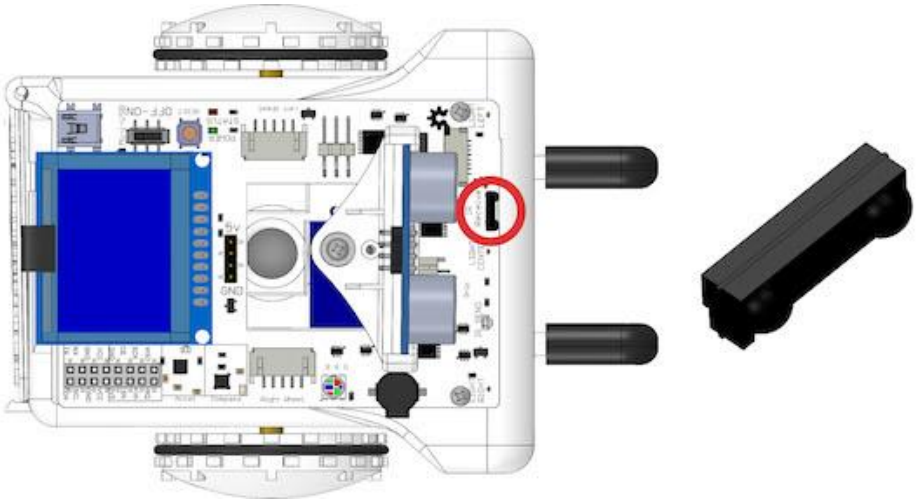
SparkiDuino already has code examples for you to use:

File > Examples > Infrared_Reflectance_Sensor

```
/*****************************************
  IR Sensors

  Sparki has sensors underneath that
  sense objects in the way, and differences
  in the surface color. This example shows
  the numbers from those on the LCD. Try
  seeing what surfaces affect the numbers.
*****************************************/
#include <Sparki.h>

void setup()
{
}

void loop()
{
  sparki.clearLCD(); // wipe the screen

  int edgeLeft   = sparki.edgeLeft();   // measure the left edge IR sensor
  int lineLeft   = sparki.lineLeft();   // measure the left IR sensor
  int lineCenter = sparki.lineCenter(); // measure the center IR sensor
  int lineRight  = sparki.lineRight();  // measure the right IR sensor
  int edgeRight  = sparki.edgeRight();  // measure the right edge IR sensor

  // write the measurements to the screen
  sparki.print("Edge Left:   ");
  sparki.println(edgeLeft);

  sparki.println();

  sparki.print("Line Left:   ");
  sparki.println(lineLeft);

  sparki.print("Line Center: ");
  sparki.println(lineCenter);

  sparki.print("Line Right:  ");
  sparki.println(lineRight);

  sparki.println();

  sparki.print("Edge Right:  ");
  sparki.println(edgeRight);

  sparki.updateLCD(); // display all of the information written to the screen
  delay(100);
}
```

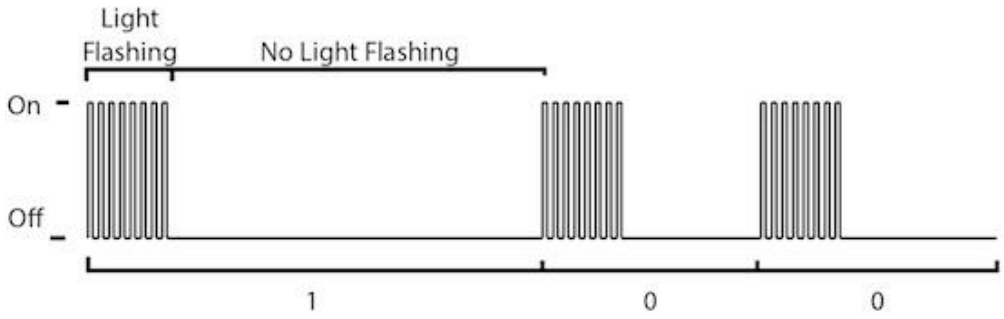
Infrared Remote Receiver

Sparki has one infrared remote receiver that can receive commands by rapidly blinking infrared lights, like those with Sparki's remote, or TV remotes.



How it works?

Infrared remotes work by sending codes via flashing lights. These lights flash really fast; Sparki's infrared LED flashes 38,000 times a second. By changing the amount of time between each flash, Sparki can send a burst that is meant to be a 1 or a 0, known as bits.



Sparki sends these flashes until an entire code has been sent. Specifically, Sparki uses the NEC infrared code protocol[4].

Using the part

With the basic Sparki code in place, you can measure an infrared sensor by using this command:

```
sparki.readIR();
```

This command returns the last code that was sent to Sparki. It returns this number in the form of an integer. If there has not been a code sent since the last time Sparki ran this command, it will send back a -1.

[4] http://techdocs.altium.com/search/wikinode/NEC+Infrared+Transmission+Protocol%20type:wikipage

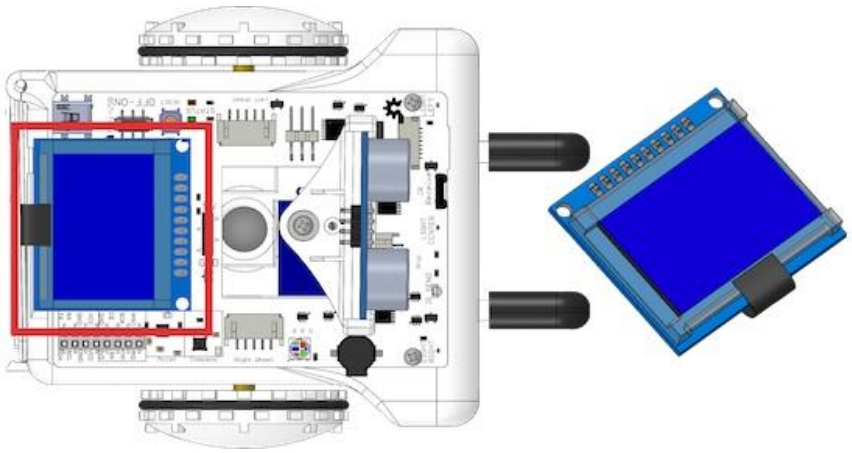
SparkiDuino already has code examples for you to use:

File > Examples > Remote

```
/*****************************************
  IR Remote

  Sparki has a sensor that lets it receives
  commands from the included remote control.
  Try moving it around!
*****************************************/
#include <Sparki.h> // include the sparki library

void setup()
{
  sparki.clearLCD();
}

void loop()
{
  int code = sparki.readIR();

  if(code != -1){
    sparki.print("Received code: ");
    sparki.println(code);
  }

  switch(code){

  // Movement buttons
  case 70: sparki.moveForward(); break;
  case 21: sparki.moveBackward(); break;
  case 67:
  case 71: sparki.moveRight(); break;
  case 69:
  case 68: sparki.moveLeft(); break;
  case 64: sparki.moveStop();
           sparki.gripperStop();
           break;

  // Gripper Buttons
  case 9:  sparki.gripperOpen(); break;
  case 7:  sparki.gripperClose(); break;

  // buzzer
  case 74: sparki.beep(); break;

  // Servo Buttons
  case 90: sparki.servo(SERVO_LEFT); break;
  case 28: sparki.servo(SERVO_CENTER); break;
  case 8: sparki.servo(SERVO_RIGHT); break;

  // RGB LED
  case 25: sparki.RGB(RGB_OFF); break;
  case 12: sparki.RGB(RGB_RED); break;
  case 24: sparki.RGB(RGB_GREEN); break;
  case 94: sparki.RGB(RGB_BLUE); break;

  default:
    break;
  }

  sparki.updateLCD();
}
```

LCD

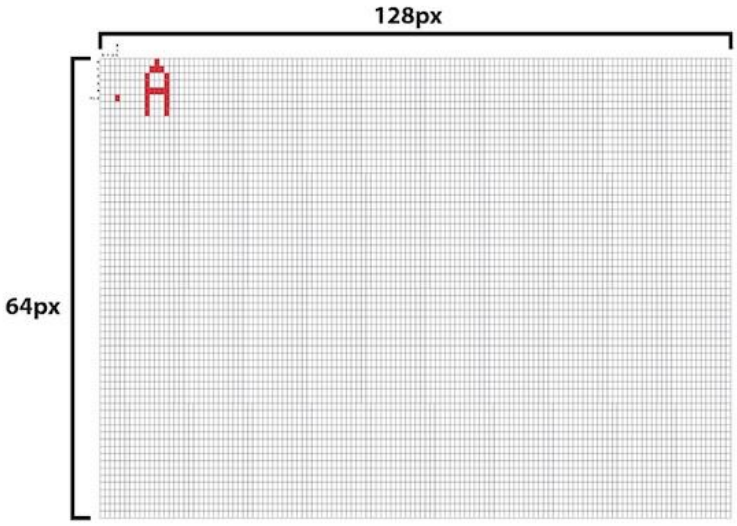
Sparki has an LCD with 128 by 64 pixels. It is used to display information in the way a computer monitor displays information.



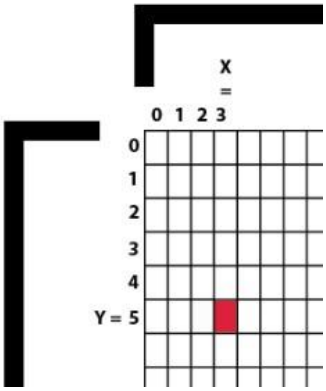
How it works?

The LCD works by having a lot of pixels, or small rectangle blocks. Pixels make up any image you see on almost any screen, whether it is a TV, computer or phone. By putting together these blocks and turning them on and off at the right time and in the right pattern, you can create images.

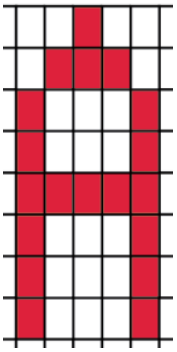
For the sake of being able to identify each pixel on Sparki's LCD, we have given co-ordinates to each pixel so it can be identified.



This pixel, for example, would be at X = 3, Y = 5:



Turn on and off enough of these pixels, and you can get an image, like the letter ‘A':



Programming each pixel on and off individually to create images would be really boring, so the computer takes ‘higher-level’ commands and turns them into pixels automatically. Instead of telling the computer to turn on and off pixels to make the letter ‘A’ for example, you can just give it the command to display ‘A’, and it will automatically display ‘A’, because someone has already told the computer how to turn on and off pixels to make an ‘A’. The faster it can do this, the smoother motion appears.

Using the part

Here are some examples of pixels with their corresponding coordinates:

- The pixel in the upper-left corner is (0, 0).
- The pixel in the first left-right row and in the last top-bottom column is (127, 0).
- The pixel in the last left-right row and in the first top-bottom column is (0, 63).
- The pixel in the last left-right row and in the last top-bottom column is (127, 63).

Try entering different coordinates for a bunch of pixels. You may want to get used to the coordinate system to the point where you can identify if a pixel will appear on the upper half or the lower half of the screen as well as knowing if it will be on the right or left hand side just by taking a look at the pixel coordinates.

Left or Right? It might be easiest to take a look at the width of the LCD screen. The middle of the screen (in regards to left and right) is at pixel number 64. So if the first pixel coordinate is higher than the number 64 then the number will appear on the right hand side of the screen, if it is lower than the number 64 it will appear on the left hand side of the screen.

Upper or Lower? Now let’s look at the height of the pixel screen. The middle of the screen (in regards to up and down) is at pixel 32. This means that if the second pixel coordinate is higher than 32 it will appear on the bottom of the LCD screen. If the second pixel coordinate is lower than 32 it will appear on the top of the LCD screen.

Actually Making a Pixel- Use the command `drawPixel( );` to draw a single pixel. For example, the command `drawPixel(3,5);` will draw a pixel at the coordinates we talked about in the beginning of this section- an X value of 3 and a Y value of 5.

High level commands

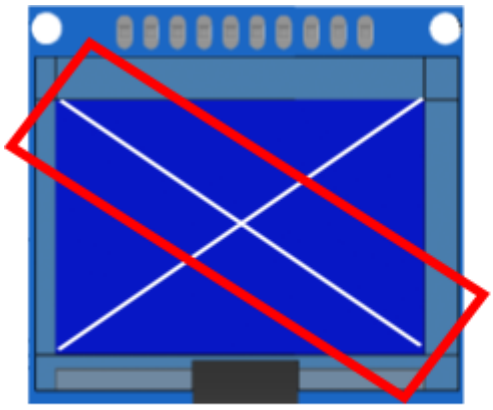
Pixels are a small part of the screen. If you were to write a letter, or draw a circle, or almost anything else by turning on and off each pixel it needed, that would be a LOT of work! That’s why even though we could still write (and read) individual pixels, we will not be doing that most of the time. Instead, we will be using simpler Sparki commands to draw things, without having to turn on and off each pixel one by one. Imagine the number of lines of code you would need to write in order to create a square that is 20 pixels by 20 pixels? It would take 400 lines of code just to make a simple square!

Let's start with a small example: drawing some lines to make a large X on the display, corner to corner.

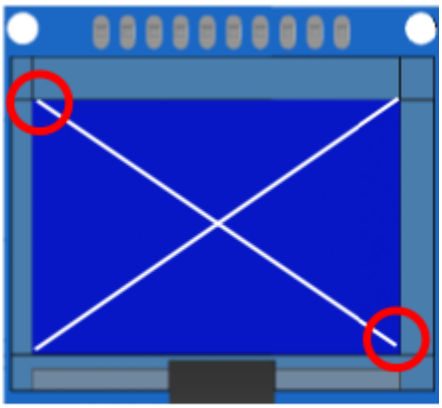
```
#include <Sparki.h>  // include the sparki library

void setup()
{
}

void loop()
{
  sparki.clearLCD(); // wipe the LCD clear

  sparki.drawLine(0,0, 127,63);
  sparki.drawLine(0,63, 127,0);

  sparki.updateLCD(); // put the drawings on the screen
  delay(1000);
}
```

So, how does this work? Let's take a general look at the code:

- First, everything is written in the loop. Why? Because the display needs to be updated from time to time. As you can see, there is a `delay(1000)` instruction at the end of the program. This means that the command that updates the display (the `sparki.updateLCD()` command) doesn't need to be executed very often. It just needs to be there when you want the display to be drawn again. Without the `sparki.updateLCD()` command the new information would never make its way onto the LCD screen. We will see better examples of this later on in the lesson.
- Second, before we can effectively draw the lines, we have to wipe the LCD clear using the `sparki.clearLCD()` instruction. If you don't do this then Sparki will leave the previous drawing on the LCD and just draw the new lines over it. That isn't a big deal for our two lines, but if we wanted to change the position of the lines then without the `sparki.clearLCD()` instruction we could draw a new line but the old line would stay on the screen. (We'll take a look at `sparki.clearLCD()` a little more in the animation section of this lesson.)
- Finally, in the middle of this program are the line drawing instructions themselves. As you can see, the `sparki.drawLine` command receives 4 (integer) numbers as parameters. What are these numbers? They are the coordinates belonging to the "important" pixels of our lines. By important pixels we mean the pixels at the beginning and at the beginning and end of each line. Let's take a look at the first line of code- `sparki.drawline(0,0, 127, 63);` The first two numbers (0, 0) are located at the upper left corner of the LCD screen and the second set of numbers (127, 63) are located at the lower right corner of the LCD screen. Sparki fills in the pixels that connect these two dots in a straight line.



Line from the first line of code



"Important" pixels

So what about circles, rectangles, or even filled zones? Let's try the following code, and then you can experiment a bit with the coordinates and dimensions of the different elements drawn there.

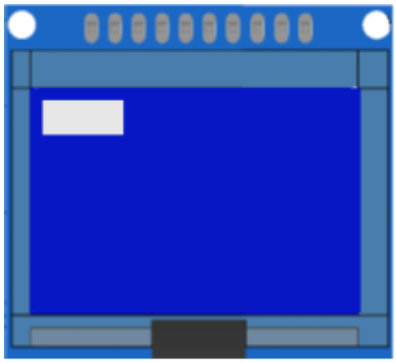
```
#include <Sparki.h>  // include the sparki library

void setup()
{

}

void loop()
{
  sparki.clearLCD();

  sparki.drawRect(5,5, 30,10);
  sparki.drawRectFilled(15,17, 30,10);

  sparki.drawCircle(55,30, 5);
  sparki.drawCircle(20,45, 12);

  sparki.drawCircleFilled(90,40, 20);

  sparki.updateLCD();
  delay(1000);
}
```

Of course, you can always go to the LCD menu:

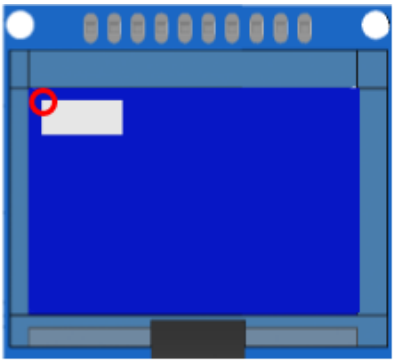
File->Examples->LCD

Rectangles - Let's take a look at how these lines of code which draw rectangles work. There are four coordinates inside the parenthesis. The first two numbers are the coordinates of one of the corners of the rectangle. The second two numbers are the coordinates of the corner that is the diagonal opposite of the first. Really, the second set of numbers could be the upper left corner and the first set of numbers could be the lower left corner of the rectangle and you could draw the same rectangle using the command:

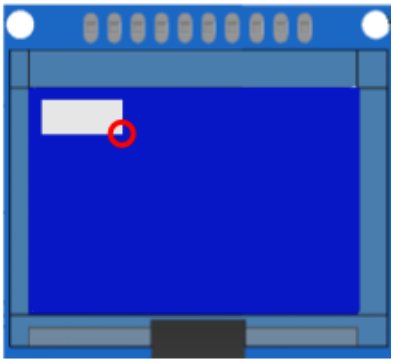
```
sparki.drawRect(30,10, 5,5);
```



drawRect(5,5, 30,10);



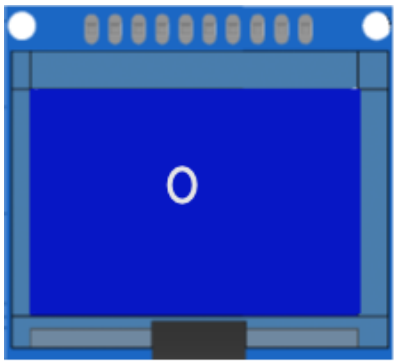
5,5 coordinate



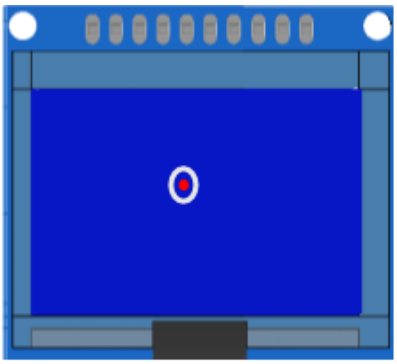
30,10 coordinate

Circles - Now let's take a look at the code that draws circles. We'll focus on the command;
`sparki.drawCircle(55, 30, 5);`

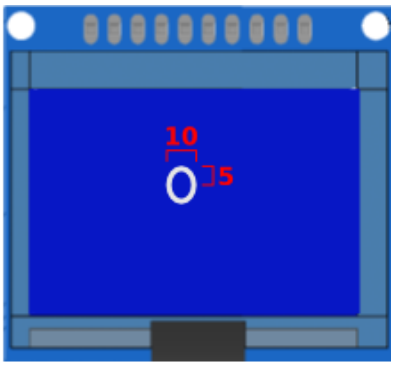
The first two numbers are the X and Y coordinates of the center of the circle Sparki is going to draw. This is exactly the same as the code for drawing a pixel. The main difference is that Sparki will draw a circle around this point. The radius of the circle is the third number, 5. So this code will draw a circle that is 10 pixels tall and 10 pixels wide with a center at 55, 30.



drawCircle(55, 30, 5);



center coordinates 55, 30



radius of 5, width of 10

Another important thing that we can do with the LCD display is writing text and numbers. This is very useful, especially once we start to work with Sparki's sensors since we can easily use the LCD to display the values of the sensors Sparki is reading. This way, if something is not working as we expect in a program using one or more sensors, we can see the exact readings on the display. You can also use the LCD to print out where Sparki is in the code you have written.

There are two ways of writing text: The first would be to write each individual letter. The easier way is just printing the text, and the Sparki LCD will automatically position each printed line after the other in the LCD. Please note that in the example below, the `sparki.println()` function can both print text strings and numbers (integers and floats).

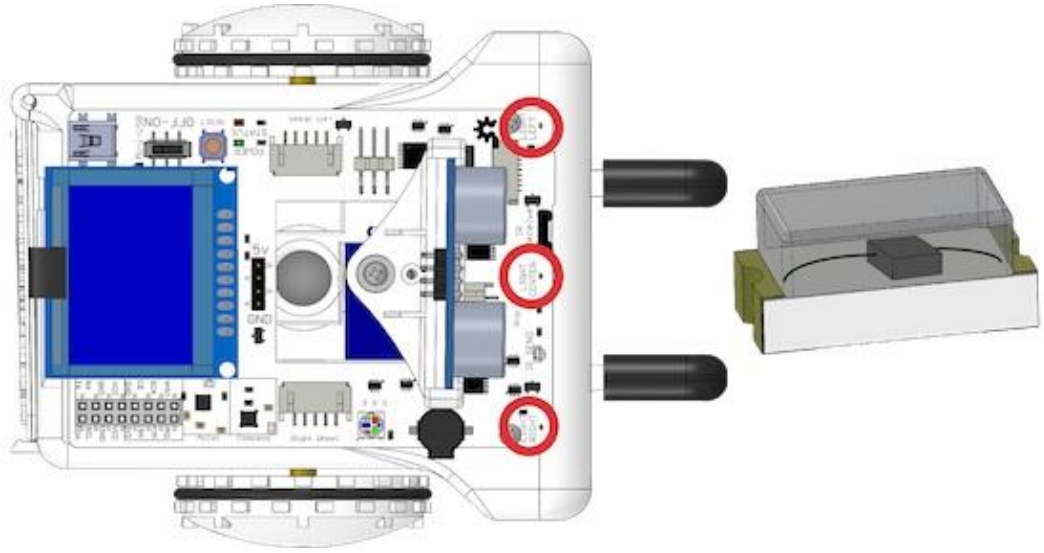
```
#include <Sparki.h>  // include the sparki library

void setup()
{

}

void loop()
{
    sparki.clearLCD(); // wipe the LCD clear
    sparki.print("abc");
    sparki.println("def");

    sparki.println(123);
    sparki.println(456.5);
    sparki.updateLCD(); // put the drawings on the screen
    delay(1000);
}
```

The other way of writing a text in the LCD is a bit trickier, but it will enable us to position a character or a text string anywhere on the display.

```
#include <Sparki.h>  // include the sparki library

void setup()
{

}

void loop()
{
    sparki.clearLCD(); // wipe the LCD clear
    sparki.drawChar(10, 1, 'a');
    sparki.drawChar(20, 2, 'b');
    sparki.drawChar(30, 3, 'c');

    sparki.drawString(40, 4, "123");
    sparki.updateLCD(); // put the drawings on the screen
    delay(1000);
}
```

Light Sensor

Sparki has three light sensors that detect the level of light that hits them.



How it works?

These sensors are phototranistors that are connected to Sparki so that the more light that hits them, the higher the voltage they send to Sparki.

Using the part

With the basic Sparki code in place, you can measure the light sensor using this command:

```
sparki.lightLeft();
sparki.lightCenter();
sparki.lightRight();
```

This command returns the amount of light in terms of an integer. The more light, the higher the number. Each sensor is slightly different, so the same amount of light may not mean the same number for each sensor.

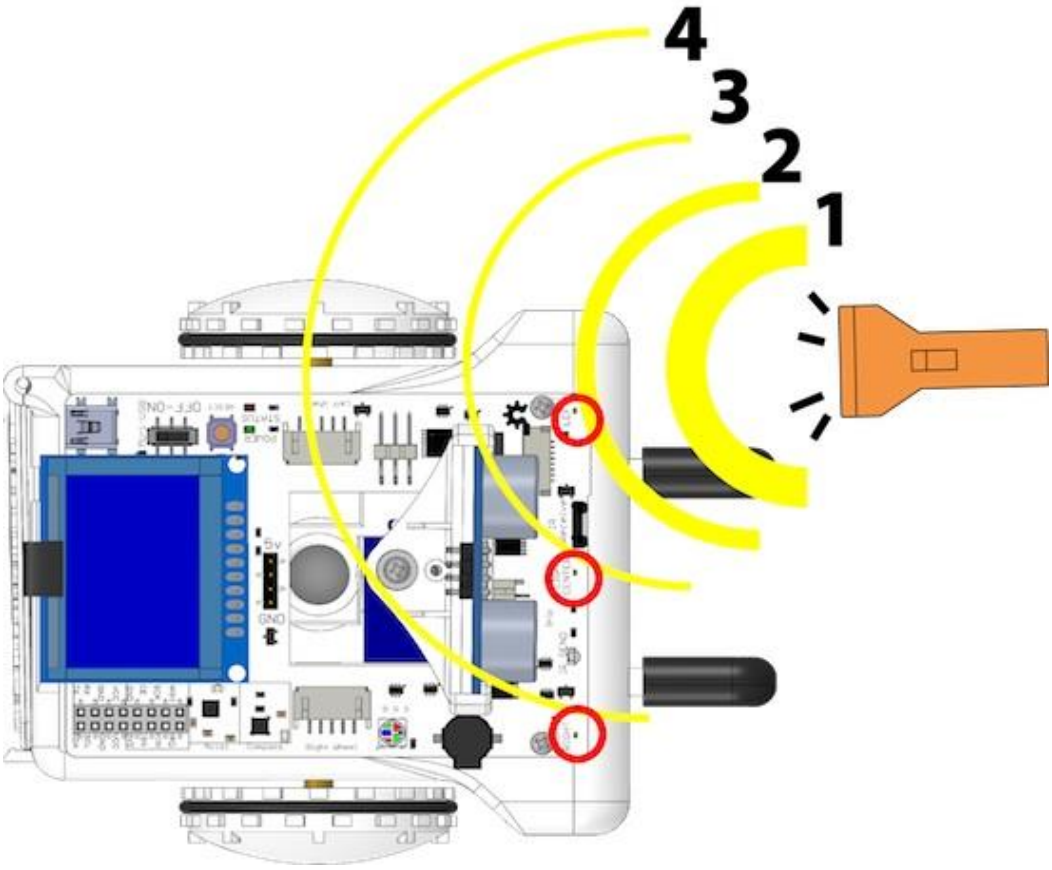
SparkiDuino already has code examples for you to use:

File->Examples->Light

```
/*****************************************
 Basic Light Sensor test

 Sparki has 3 light sensors in the front,
 one each on the left, middle and right.
 This program shows how to read the sensors
 and display the information on the LCD.
******************************************/
#include <Sparki.h> // include the robot library

void setup()
{

}

void loop()
{
  sparki.clearLCD(); // wipe the screen

  int left   = sparki.lightLeft();   // measure the left light sensor
  int center = sparki.lightCenter(); // measure the center light sensor
  int right  = sparki.lightRight();  // measure the right light sensor

  // write the measurements to the screen
  sparki.print("Left   ");
  sparki.println(left);

  sparki.print("Center ");
  sparki.println(center);

  sparki.print("Right  ");
  sparki.println(right);

  sparki.updateLCD(); // display all of the information written to the screen
  delay(100);
}
```

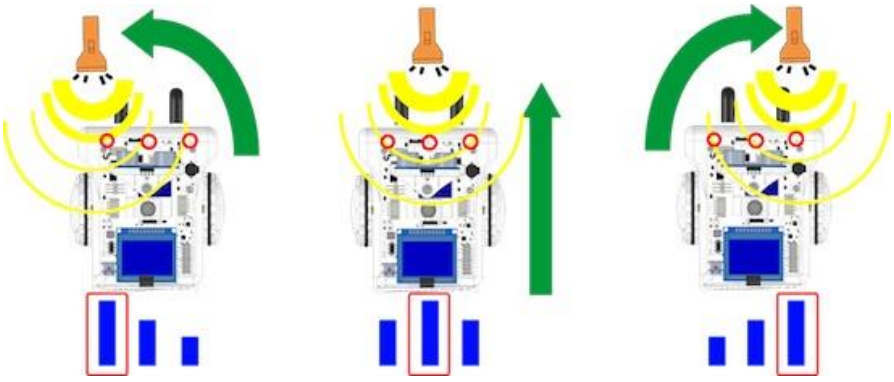
Example – Follow and Avoid Light

Each of Sparki's light sensors can measure how much light is hitting them. The idea behind this follow light example is that the light is strongest on the sensor nearest to the light. Just like being further away from someone yelling makes it quieter, being further away makes the light dimmer.



The closer the sensor is to the light, the stronger the light from the source. In this example, the light arriving at the left sensor is going to be the strongest, the center sensor will receive the middle-most light, and the right sensor is going to get the least light.

Please remember to check that the batteries are properly connected (and charged!). And as we are going to use the motors here, please check that the On/Off Switch is on. Another important thing to take care of when playing with the robot's motors is to be careful not to be working over a table. A fall from that table could permanently damage your Sparki.



We will have Sparki following light using this idea and if statements. We will conduct three if tests. Each test will determine if the sensor in that test is getting more light than either of the two other sensors. If that sensor is getting more light than the other two, it will move in that sensor's direction.

Following Light

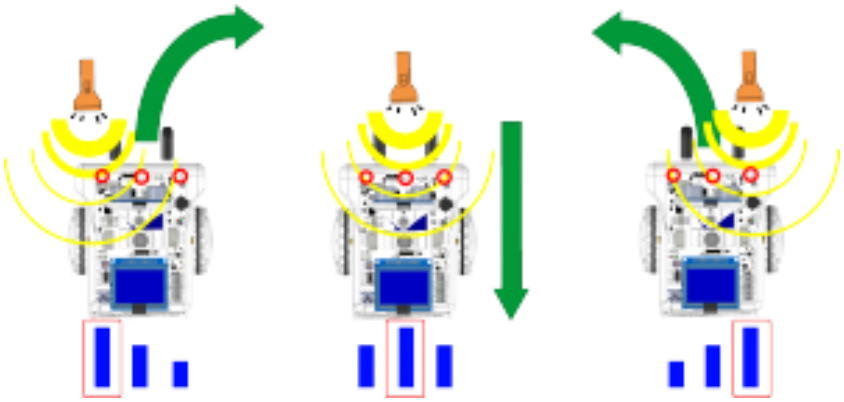
Try out the example code for light following in;

Examples->Follow_Light

```
#include <Sparki.h> // include the sparki library

void setup()
{
}

void loop()
{
  int left   = sparki.lightLeft();   // measure the left light sensor
  int center = sparki.lightCenter(); // measure the center light sensor
  int right  = sparki.lightRight();  // measure the right light sensor

  if ( (center > left) && (center > right) ){  // if the center light is the strong-
est
    sparki.moveForward(); // move Sparki Forward
  }

  if ( (left > center) && (left > right) ){    // if the left light is the strongest
    sparki.moveLeft(); // move Sparki Left
  }

  if ( (right > center) && (right > left) ){   // if the right light is the strongest
    sparki.moveRight(); // move Sparki Right
  }

  delay(100); // wait 0.1 seconds
}
```

Avoiding Light

Of course, the same logic (and nearly the same program) can be used to have Sparki avoid light instead of following it.



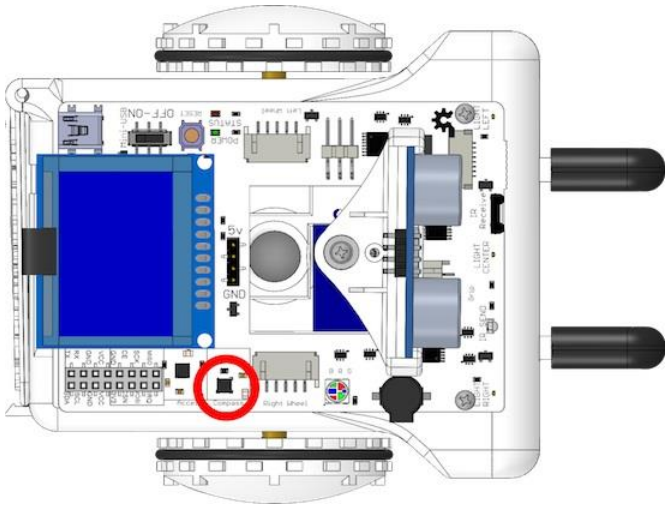
You can find the code in;

Examples->Avoid_Light

```
#include <Sparki.h> // include the sparki library

void setup()
{
}

void loop()
{
  int left   = sparki.lightLeft();   // measure the left light sensor
  int center = sparki.lightCenter(); // measure the center light sensor
  int right  = sparki.lightRight();  // measure the right light sensor

  if ( (center > left) && (center > right) ){  // if the center light is the strong-
est
    sparki.moveBackward();
  }

  if ( (left > center) && (left > right) ){    // if the left light is the strongest
    sparki.moveRight();
  }

  if ( (right > center) && (right > left) ){   // if the right light is the strongest
    sparki.moveLeft();
  }

  delay(100); // wait 0.1 seconds
}
```

Magnetometer

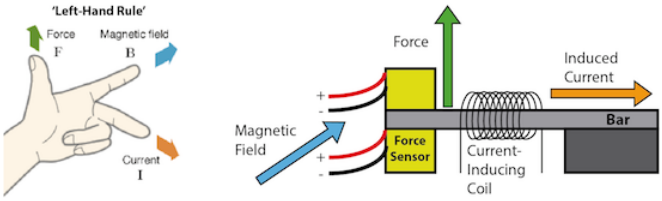
Sparki has a 3-Axis magnetometer. It is used to detect the magnetic field Sparki is experiencing. This can mean the earth's magnetic field, the field generated by Sparki's motors and wires, or many other sources. It can do this in all 3 XYZ axis, which are left/right (X), forward/backwards (Y), and up/down (Z). This is the same part that smartphones use to act as a compass.



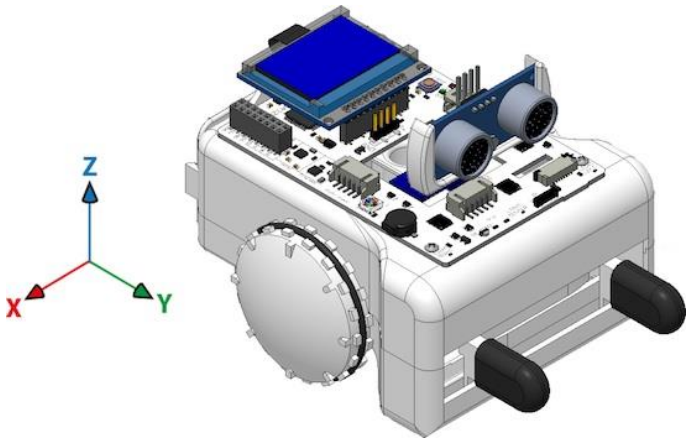
How it works?

The magnetometer is what's called a microelectromechanical, or MEMs device. MEMs are made of very tiny mechanical parts and wires too small to see, made in the same way microchips are made. Inside the part, too tiny to see, are tiny coils and a beam. Each axis has a beam. That beam has a pressure sensor on each side, and

a coil wrapped around it passing current through it. Due to Lorentz force[5], the stronger the magnetic field, the more the bar with the current passing through it bends. The pressure sensors measure this bending, do calculations on how much magnetic field it means, and send this information to Sparki.



The part has 3 different axis, left/right (X), forward/backwards (Y), and up/down (Z).



The direction of the arrows shows the positive direction of the reading. If the magnetic field is in the opposite direction, the reading will be negative.

Using the part

With the basic Sparki code in place, you can measure the magnetic field in each axis using these commands:

```
sparki.magX();
sparki.magY();
sparki.magZ();
```

This command returns the magnetic field the magnetometer reads on the X, Y or Z axis in milliGauss.

SparkiDuino already has code examples for you to use:

File > Examples > Magnetometer

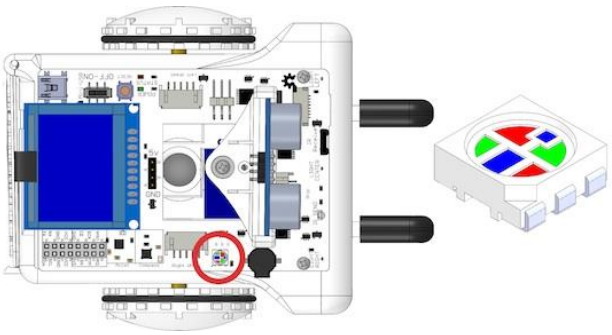
```
/*****************************************
 Basic Magnetometer Sensor test

 Sparki has a 3-Axis magnetometer. It is
 used to detect the magnetic field Sparki is
 experiencing. This can mean the earth's
 magnetic field, the field generated by
 Sparki's motors and wires, and many other
 sources.  It can do this in all 3 XYZ axis,
 which are left/right (X),
 forward/backwards (Y), and up/down (Z). This
 is the same part that smartphones use to act
 as a compass.

 This program shows how to read the sensor
 and display the information on the LCD.
*****************************************/
```

[5] http://en.wikipedia.org/wiki/Lorentz_force

```ino
#include <Sparki.h> // include the robot library

void setup()
{
}

void loop() {
  sparki.clearLCD(); // wipe the screen

  float x  = sparki.magX();   // measure the accelerometer x-axis
  float y  = sparki.magY();   // measure the accelerometer y-axis
  float z  = sparki.magZ();   // measure the accelerometer z-axis

  // write the measurements to the screen
  sparki.print("Mag X: ");
  sparki.println(x);

  sparki.print("Mag Y: ");
  sparki.println(y);

  sparki.print("Mag Z: ");
  sparki.println(z);

  sparki.updateLCD(); // display all of the information written to the screen
  delay(100);
}
```

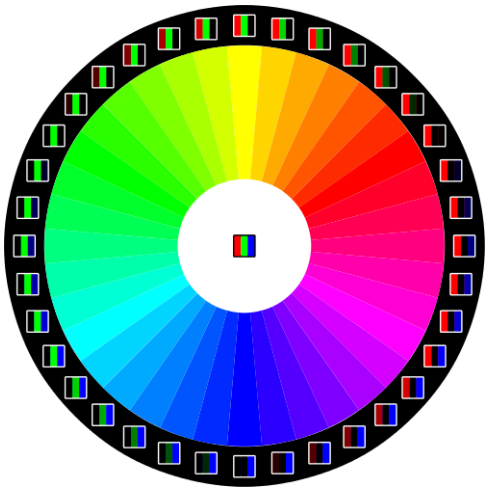
RGB LED

Sparki has an RGB (Red, Green, Blue) LED light. It is used to display any color, useful for showing simple information.



How it works?

By combining the colors red, green and blue you can make any color you want.



To use this idea, the part has a tiny red, green and blue LED light closely placed together. By adjusting the intensity of each of these LED lights, it is possible to create any color.

Using the part

With the basic Sparki code in place, you can change the RGB LED color using this command:

```
sparki.RGB(red, green, blue);
```

Where red, green and blue are numbers from 0 to 100. For example, if you wanted the light to be red-only, you would use:

```
sparki.RGB(100, 0, 0);
```

You can also use the name of a color (as long as it's all capital letters) to display it. So for the above example of red, you would use:

```
sparki.RGB(RGB_RED);
```

Sparki's code comes with the following predefined colors:

```
RGB_RED
RGB_GREEN
RGB_BLUE
RGB_WHITE
RGB_OFF
```

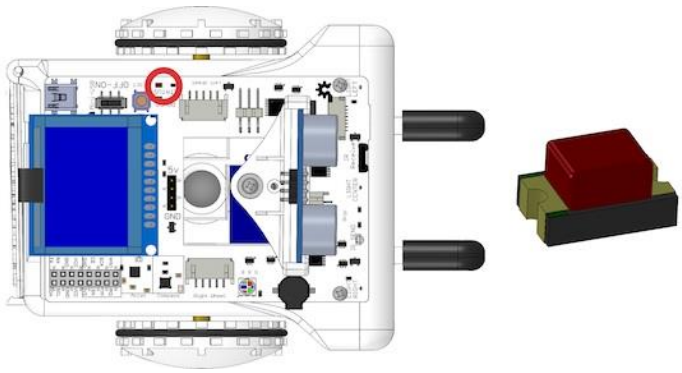
SparkiDuino already has code examples for you to use:

File->Examples->Sparki->RGB

```
/*****************************************
 Basic RGB test

 Sparki has a Red, Green, Blue LED on its
 back. Using Red, Green and Blue, you can
 make any color you want. The brightness of
 each color goes from 0 (dark) to 100 (full
 brightness). What colors can you make?

 Here are Sparki's colors:
             R    G    B
 RGB_RED     100, 0,   0
 RGB_ORANGE  90,  100, 0
 RGB_YELLOW  60,  100, 0
 RGB_GREEN   0,   100, 0
 RGB_BLUE    0,   0,   100
 RGB_PINK    90,  0,   100
 RGB_INDIGO  20,  0,   100
 RGB_VIOLET  60,  0,   100
 RGB_WHITE   60,  100, 90
 RGB_OFF     0,   0,   0

 http://arcbotics.com/products/sparki/parts/rgb-led/
*****************************************/
#include <Sparki.h> // include the sparki library

void setup()
{
}

void loop()
{
  sparki.RGB(100,0,0); // Make the LED maximum Red
  delay(500); // wait 0.5 seconds (500 milliseconds)

  sparki.RGB(0,100,0); // Make the LED maximum Green
  delay(500);

  sparki.RGB(0,0,100); // Make the LED maximum Blue
  delay(500);

  sparki.RGB(RGB_WHITE); // Make the LED white (all colors)
  delay(500);

  sparki.RGB(RGB_OFF); // Make the LED white (all colors)
  delay(500);
}
```

Status LED

Sparki has a small red LED light that can be used to display simple information. You can turn it on and off using `digitalWrite`, or partially turn it on using `analogWrite`.



Using the part

With the basic Sparki code in place, you can use the status LED using these commands. You can copy-paste them into the code:

```
digitalWrite(STATUS_LED, HIGH);  // will turn the LED on
digitalWrite(STATUS_LED, LOW);   // will turn the LED off
```

You can also use the analogWrite function to tun the LED partially on or off:

```
digitalWrite(STATUS_LED, 0);       // will turn the LED 0% on (off)
digitalWrite(STATUS_LED, “123”);   // will turn the LED 50% on
digitalWrite(STATUS_LED, “255”);   // will turn the LED 100% on
```

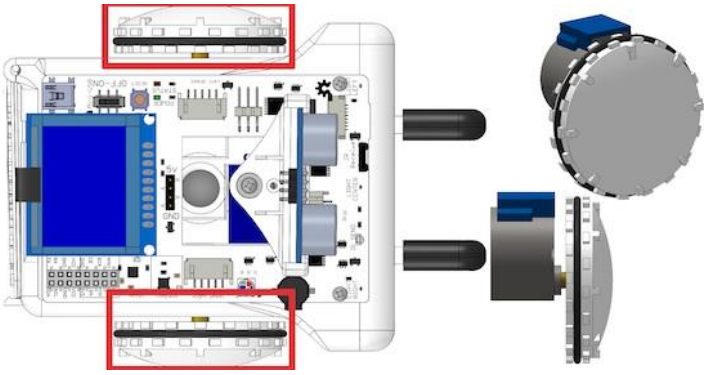
SparkiDuino already has code examples for you to use:

File > Examples > Status_LED

```
/*****************************************
 Status LED test

 Sparki has a red status LED, next to the
 power on LED. This code first blinks it
 on then off, then fades it on then off.
*****************************************/
#include <Sparki.h>  // include the sparki library
void setup()
{
}
void loop()
{
    digitalWrite(STATUS_LED, HIGH); // will turn the LED on
    delay(500); // wait 0.5 seconds
    digitalWrite(STATUS_LED, LOW); // will turn the LED off
    delay(500); // wait 0.5 seconds

    for(int i=0; i<255; i++) // pulse the LED on
    {
      analogWrite(STATUS_LED, i);
      delay(5);
    }

    for(int i=255; i>0; i--) // pulse the LED off
    {
      analogWrite(STATUS_LED, i); // will turn the LED 100% on
      delay(5);
    }
}
```

Wheels

Sparki has two wheels that it uses to move around with.



How it works?

Sparki's two wheels are driven by two geared stepper motors[6]. This special type of motor lets you command Sparki to move a precise very distance.

Using the part

With the basic Sparki code in place, you can move Sparki using these commands:

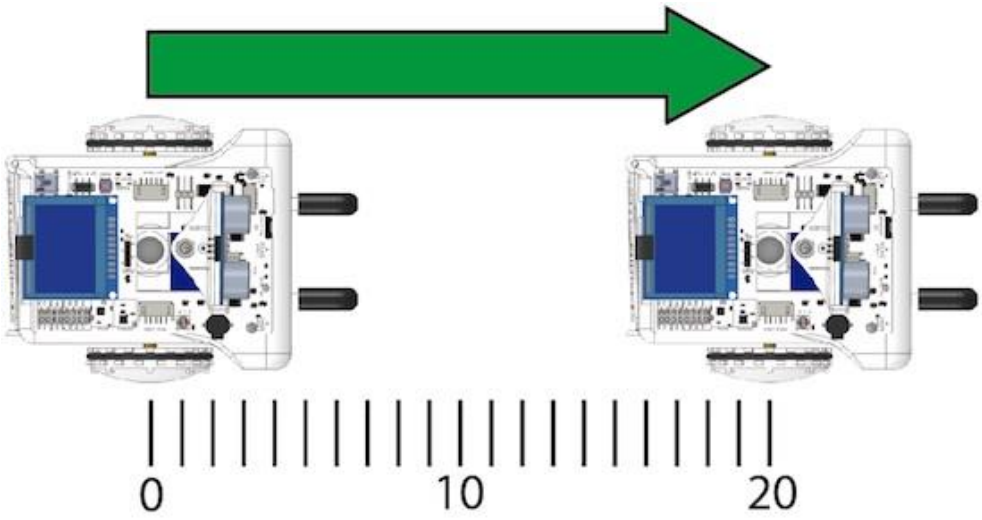
```
sparki.moveForward();
sparki.moveBackward();
sparki.moveLeft();
sparki.moveRight();

sparki.moveStop();
```

Each of these commands move Sparki in a certain direction. Sparki will continue to drive in that direction until a `moveStop` command is run.

You can also tell Sparki to move a certain amount with each command. For the `moveForward` and `moveBackward` commands, the number is in centimeters. For `rotateLeft` and `rotateRight`, it is in degrees. Sparki runs this command until it is done, then performs the next command.

For example, to move forward 20 centimeters and then stop, you would use:

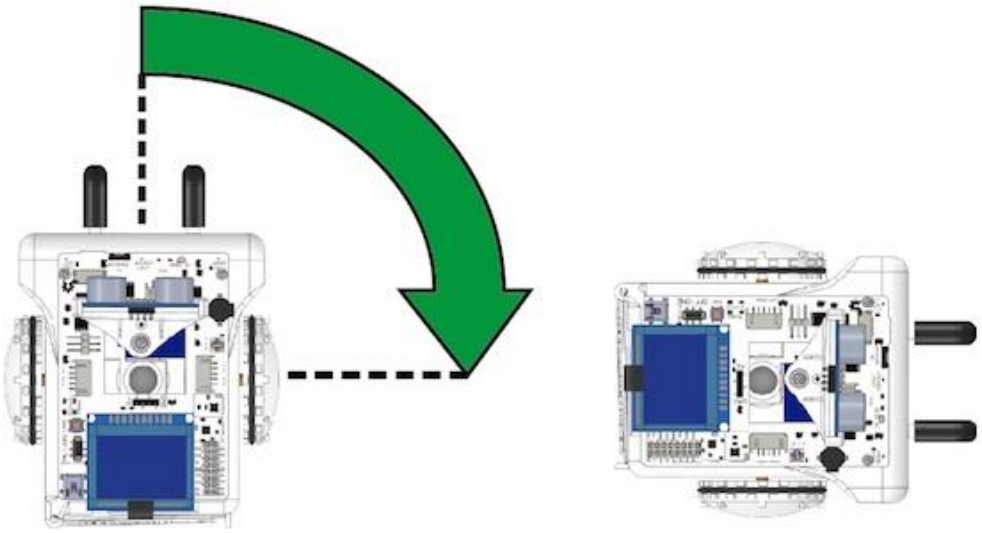
```
sparki.moveForward(20);
```



[6] http://arcbotics.com/parts/stepper-motor/

To rotate right 90 degrees and then stop, you would use:

```
sparki.moveRight(90);
```



SparkiDuino already has code examples for you to use:

File > Examples > Sparki > Wheels

```
/*****************************************
 Basic motor test

 Move Sparki forward and backwards, then
 rotate right and left, then stop. Repeat!
 Can you get Sparki to do any cool motions?
 Maybe draw something by sticking a pen down
 the center?
*****************************************/
#include <Sparki.h> // include the robot library

void setup()
{
}

void loop()
{
  sparki.moveForward();// move the robot forward
  delay(1000); // wait a second (1000 milliseconds)

  sparki.moveBackward(); // move the robot backward
  delay(1000);

  sparki.moveRight(); // rotate the robot clockwise
  delay(1000);

  sparki.moveLeft(); // rotate rhe robot counter-clockwise
  delay(1000);

  sparki.moveStop(); // stop all robot wheels
  delay(2000); // wait two seconds (2000 milliseconds)
}
```

And what about controlling the individual speed of each wheel? We can go a little deeper and use some lower level instructions, ones that are close to what Spark is actually doing, but require more of them to get things done.

```
sparki.motorRotate(int motor, int direction, int speed);
sparki.motorStop(int motor);
```

Where:

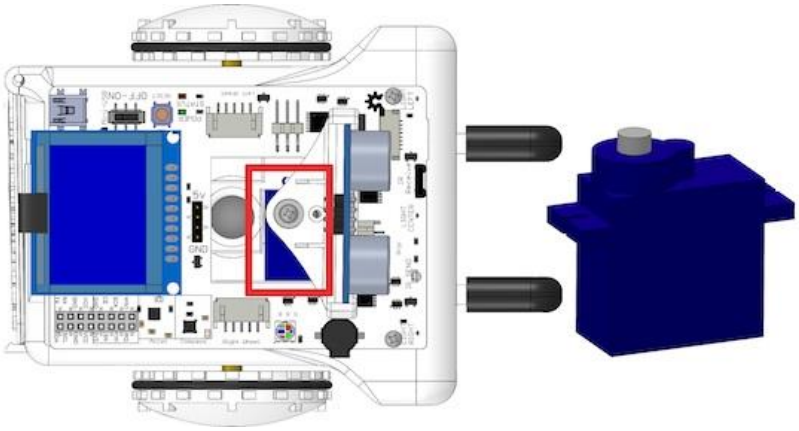
- **Motor** is the motor number, which can be MOTOR_LEFT or MOTOR_RIGHT (Tip: you can also try to use this instruction to move the gripper, using the value MOTOR_GRIPPER as the motor parameter).
- **Direction** indicates the direction of rotation for that motor, which can be DIR_CW (clockwise) or DIR_CCW (counterclockwise).
- **Speed** is a number that indicates, as a percentage, the speed of rotation. Thus, its value ranges from 0 to 100.

Here is a small example, to make the robot pivot around an external center of rotation:

```
#include <Sparki.h> // include the sparki library

void setup()
{
}

void loop()
{
    sparki.motorRotate(MOTOR_LEFT, DIR_CCW, 50);
    sparki.motorRotate(MOTOR_RIGHT, DIR_CW, 100);

    delay(4000); // wait a second (1000 milliseconds)

    sparki.motorRotate(MOTOR_LEFT, DIR_CCW, 100);
    sparki.motorRotate(MOTOR_RIGHT, DIR_CW, 50);

    delay(4000);

    sparki.motorStop(MOTOR_LEFT);
    sparki.motorStop(MOTOR_RIGHT);
}
```

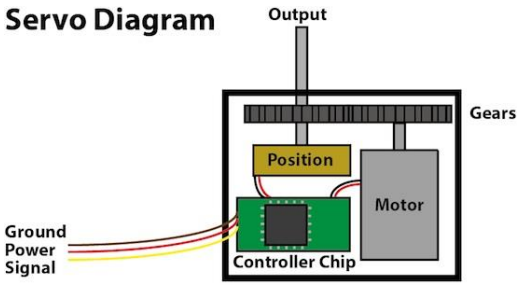
Servo

Sparki has a servo motor that moves its head around.



How it works?

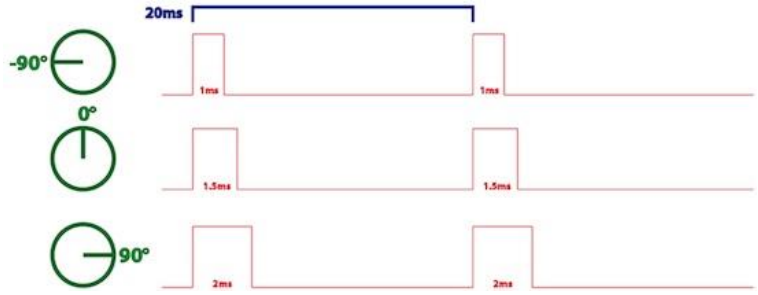
Servo motors are special types of motors that can go to the exact position that you tell them to. They can do this because in addition to the motor, it has a sensor that measures the position of the motor, and a chip that uses the position information from the sensor to control the motor.



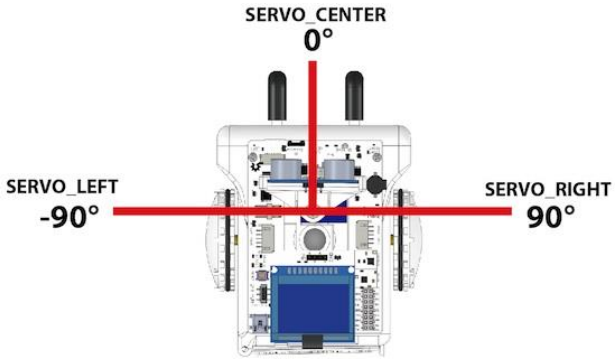
When the chip detects from the position sensor that the position is not where it is supposed to be, it tells the motor to move towards the correct position until it reaches it.

You can tell the chip what position it is supposed to be by sending it a special signal. Don't worry, Sparki's servo code below already takes care of sending this special signal. All you have to tell it is what angle!

50 times a second (or every 20 milliseconds), this signal sends an on-off pulse. The length of this on-off pulse tells the servo what position it should be in. If it is 1000 ms, then it should be 90 left. If it is 1.5 milliseconds, then the servo should be centered. If it is 2500ms, then it should be 90 degrees right.



For Sparki, the angles of the servo has been shown on the following diagram.



Using the part

With the basic Sparki code in place, you can tell the servo to move using these commands:

```
sparki.servo(degree);
```

You can use any number from -90 to 90 in place of degree. This is equivalent to looking left and looking right. You can also use these keywords (in all caps).

```
SERVO_LEFT
SERVO_CENTER
SERVO_RIGHT
```

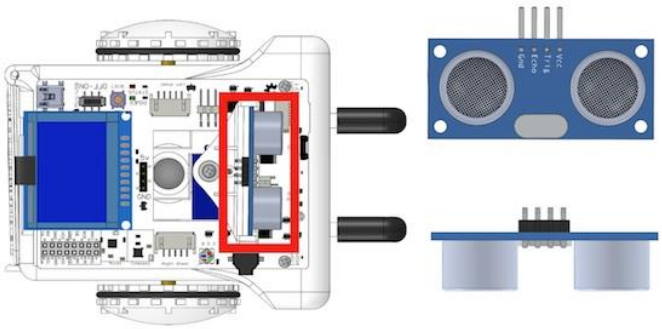
SparkiDuino already has code examples for you to use:

File > Examples > Servo

```
/*****************************************
 Basic Servo test

 Move the servo on Sparki's head. The servo
 can rotate from -90 to 90 degrees. -90 is
 facing left, 0 is facing forward,  and 90
 is facing right.
*****************************************/
#include <Sparki.h> // include the sparki library

void setup()
{
}
void loop()
{
  sparki.servo(SERVO_LEFT); // rotate the servo to is -90 degree postion (left)
  delay(1000);

  sparki.servo(SERVO_CENTER); // rotate the servo to is 0 degree postion (forward)
  delay(1000);

  sparki.servo(SERVO_RIGHT); // rotate the servo to is 90 degree postion (right)
  delay(1000);

  sparki.servo(-90); // rotate the servo to is -90 degree postion (left)
  delay(1000);

  sparki.servo(0); // rotate the servo to is 0 degree postion (forward)
  delay(1000);

  sparki.servo(90); // rotate the servo to is 90 degree postion (right)
  delay(1000);
}
```

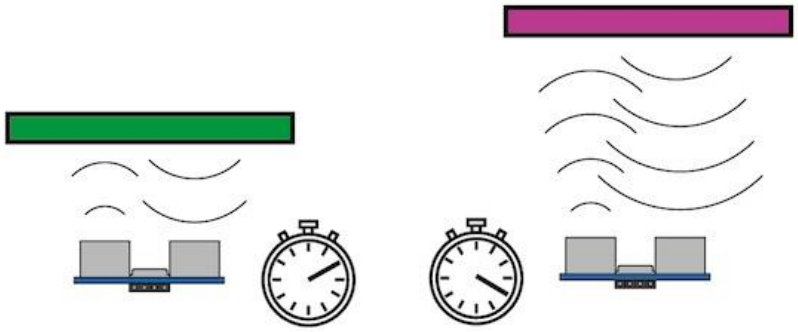
Ultrasonic Range Finder

Sparki has an Ultrasonic Range Finder that measures distance with sound waves like a wireless sonic ruler.

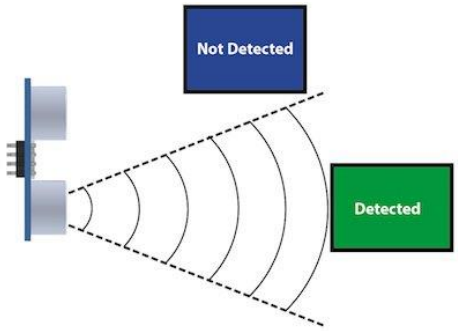


How it works?

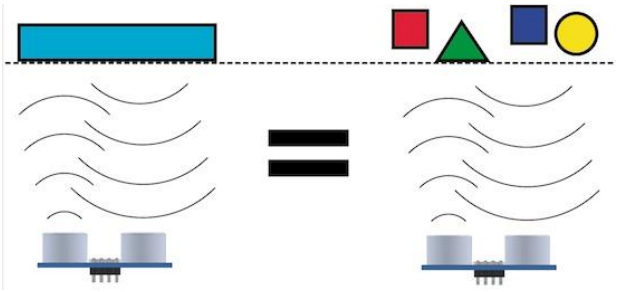
This sensor finds objects with sound the same way a bat or dolphin does. One part of the sensor is a speaker that sends out a sound wave. The other part is a microphone that then measures how long it takes for the sound wave to come back. The longer it takes to come back, the further away the object.



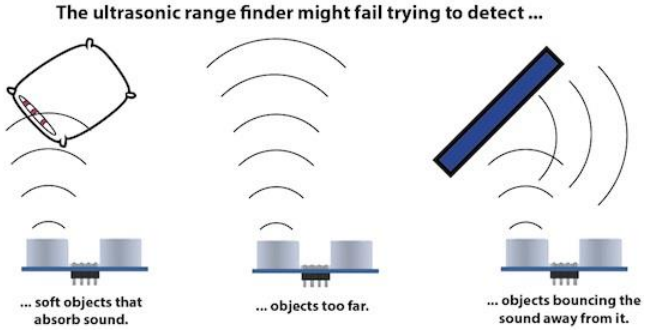
Measure the time it takes for the sound to come back, and you can tell the distance. The speaker sends out the sound in a cone, so only objects within the cone will return sound and be detected. Objects in the center of the cone do best.



This sensor is a simple ultrasonic rangefinder. It cannot detect individual objects. It can only tell you how long it took the sound sent out to return. It will measure the closest object.



If the object absorbs sound, or if there is nothing for a long distance, or if the object bounces the sound at an angle, there might not be enough sound returning to measure anything.



Using the part

With the basic Sparki code in place, you can measure an infrared sensor by using this command:

```
sparki.ping();
```

This command returns the distance the sensor reads in centimeters in the form of a float.

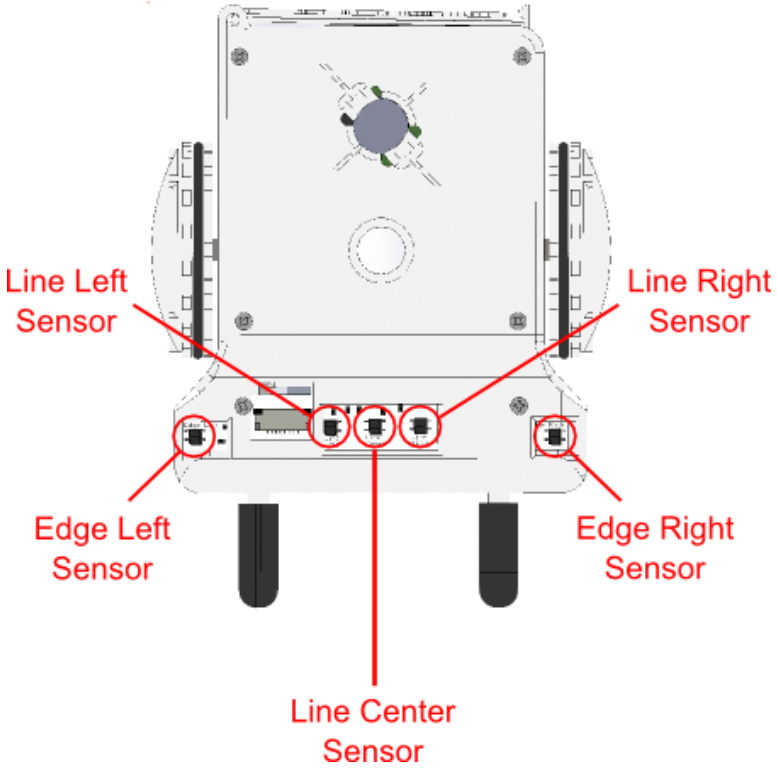
SparkiDuino already has code examples for you to use:

File > Examples->Ultrasonic_Range_Finder

```
/*****************************************
 Basic Ultrasonic test

 Show the distance Sparki's eyes are reading
 on the LCD. Sparki will beep when something
 is too close. If it measures -1, that means
 the sensor is either too close or too far
 from an object
*****************************************/
#include <Sparki.h> // include the robot library

void setup()
{
}

void loop()
{
    sparki.clearLCD();

    int cm = sparki.ping(); // measures the distance with Sparki's eyes

    sparki.print("Distance: ");
    sparki.print(cm); // tells the distance to the computer
    sparki.println(" cm");

    if(cm != -1) // make sure its not too close or too far
    {
        if(cm < 10) // if the distance measured is less than 10 centimeters
        {
            sparki.beep(); // beep!
        }
    }

    sparki.updateLCD();
    delay(100); // wait 0.1 seconds (100 milliseconds)
}
```

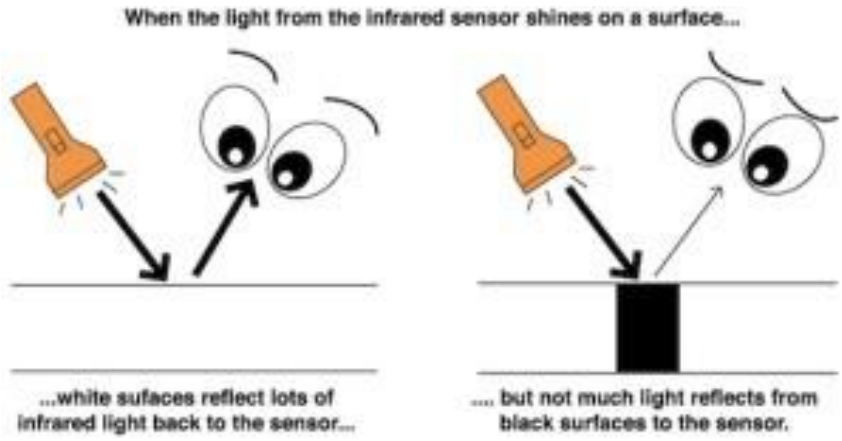
Line Following

Sparki can use the sensors below the hands to sense edges and lines. In this tutorial you'll have Sparki follow a line. Sparki has an array of infrared sensors underneath that it uses to detect if there are lines below it. We're going to program Sparki to follow the dark lines in our path, since they are printed over a more reflective background (typically, the path is printed in dark black, over a white background).



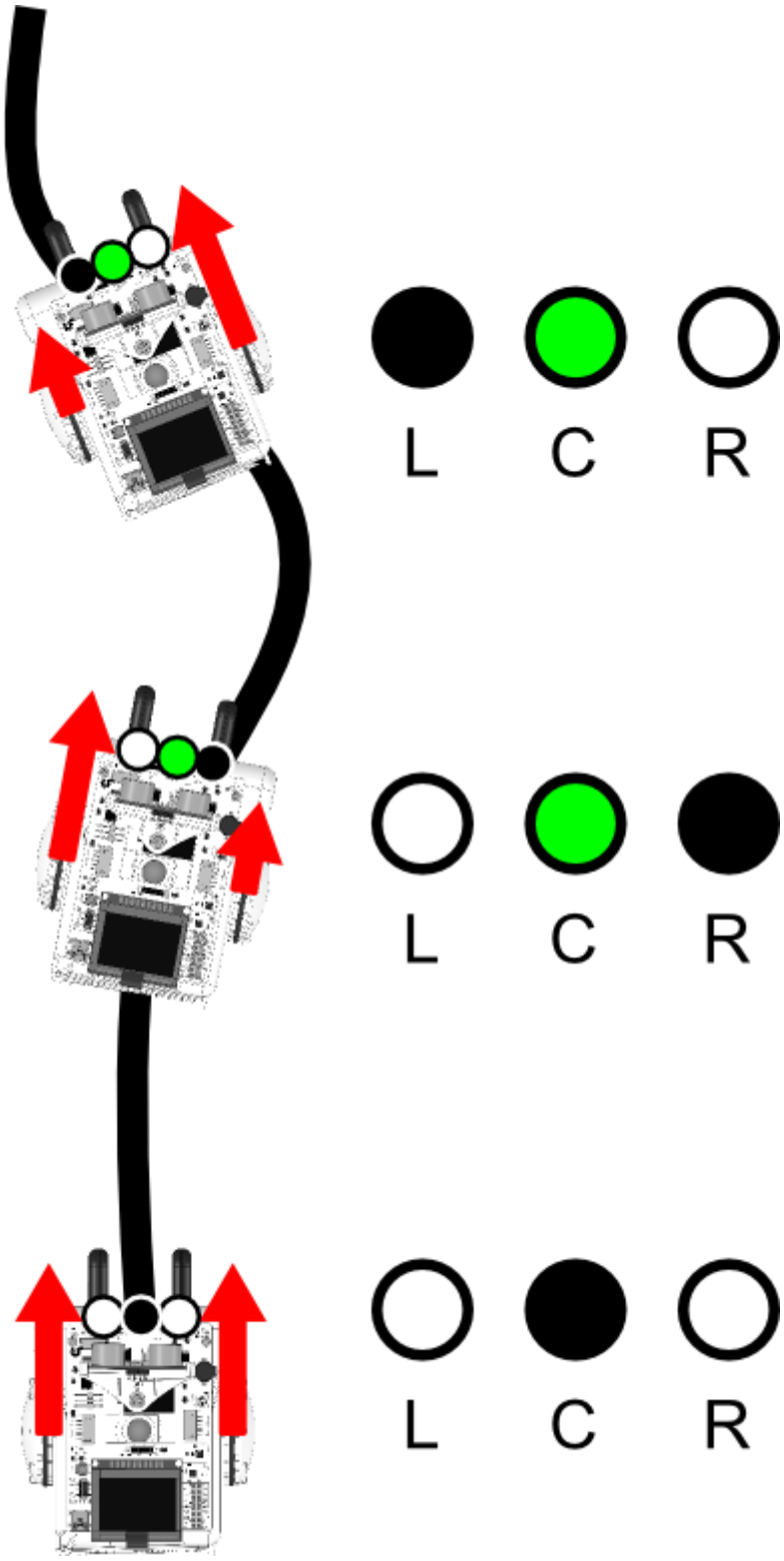
Infrared reflectance sensors can be thought of as an infrared light (which humans can not see), and some special eyes that look at that light. The eyes are made out of a special photo sensing material and an infrared LED (or Light Emitting Diode):

Here are the parts again, but with pictures showing what you'd normally think of them as:



In the following code part Sparki continues to move forward as long as it detects a line underneath the center sensor. If there is no line underneath it, it will look under the left and right line sensors to see if it has started to move off course. If there is a line underneath the left sensor, it will move left to move the robot back to the line (by accelerating the right wheel). If there is a line underneath the right sensor, it will move right to move the robot back to the line.

The following image shows the possible states of these sensors, the different ways they can be thought of, as seen by our program. The black dot means that the sensor is reading a line below it. A white dot means that the white background is below the sensor. With the green dots means Sparki does not care what color it is.



This code requires integers, variables, basic programming, wheel code, if statements and infrared reflector sensors. The readings from the infrared sensors will be sent to the LCD display so we can have a better idea of what is happening inside Sparki's brain:

```
/*****************************************************
 Sparki Line-following example

 Threshold is the value that helps you
 determine what's black and white. Sparki's
 infrared reflectance sensors indicate white
 as close to 900, and black as around 200.
 This example uses a threshold of 500 for
 the example, but if you have a narrow line,
 or perhaps a lighter black, you may need to adjust.
*****************************************************/

#include <Sparki.h> // include the sparki library

void setup()
{
}

void loop() {
  int threshold = 500;

  int lineLeft   = sparki.lineLeft();   // measure the left IR sensor
  int lineCenter = sparki.lineCenter(); // measure the center IR sensor
  int lineRight  = sparki.lineRight();  // measure the right IR sensor

  if ( lineLeft < threshold ) // if line is below left line sensor
  {
    sparki.moveLeft(); // turn left
  }

  if ( lineRight < threshold ) // if line is below right line sensor
  {
    sparki.moveRight(); // turn right
  }

  // if the center line sensor is the only one reading a line
  if ( (lineCenter < threshold) && (lineLeft > threshold) && (lineRight > thres-
hold) )
  {
    sparki.moveForward(); // move forward
  }

  sparki.clearLCD(); // wipe the screen

  sparki.print("Line Left: "); // show left line sensor on screen
  sparki.println(lineLeft);

  sparki.print("Line Center: "); // show center line sensor on screen
  sparki.println(lineCenter);

  sparki.print("Line Right: "); // show right line sensor on screen
  sparki.println(lineRight);

  sparki.updateLCD(); // display all of the information written to the screen

  delay(100); // wait 0.1 seconds
}
onds)
}
```

Parts of Sparki

This is a list for every actuator and sensor on Sparki, with a quick reminder reference below it for the commands that go along with it.

Accelerometer

```
sparki.accelX();
sparki.accelY();
sparki.accelZ();
```

Bluetooth Module

```
Serial1.read();
Serial1.print();
Serial1.println();
```

Buzzer

```
sparki.beep();
sparki.(int freq);
sparki. .(int freq, int time);
```

Expansion Port

Gripper

```
sparki.gripperOpen();
sparki.gripperOpen(width_cm);
sparki.gripperClose();
sparki.gripperClose(width_cm);
sparki.gripperStop();
```

Infrared LED

```
sparki.sendIR();
```

Infrared Reflectance Sensor

```
sparki.edgeLeft();
sparki.lineLeft();
sparki.lineCenter();
sparki.lineCenter();
sparki.edgeRight();
```

Infrared Remote

Infrared Remote Receiver

```
sparki.readIR();
```

LCD

```
sparki.clearLCD();
sparki.updateLCD();
sparki.drawPixel(xPixel, yPixel);
sparki.readPixel(xPixel, yPixel);
sparki.drawChar(xPixel, yLine, char);
sparki.drawString(xPixel, yLine, *char);
sparki.drawLine(xStart, yStart, xEnd, yEnd);
sparki.drawRect(xCenter, yCenter, width, height);
sparki.drawRectFilled(xCenter, yCenter, width, height);
sparki.drawCircle(xCenter, yCenter, radius);
sparki.drawCircleFilled(xCenter, yCenter, radius);
sparki.drawBitmap(xStart, yStart, *bitmap, width, height);
```

Light Sensor

```
sparki.lightLeft();
sparki.lightCenter();
sparki.lightRight();
```

Magnetometer
```
sparki.magX();
sparki.magY();
sparki.magZ();
```

Marker Holder

On/Off Switch

Radio
```
radio.writeChar(char);
radio.writeInt(int);
radio.writeFloat(float);
radio.readChar();
radio.readInt();
radio.readFloat();
radio.available();
```

Reset Button

RGB LED
```
sparki.RGB(R, G, B);
```

Status LED
```
digitalWrite(STATUS_LED, HIGH);
digitalWrite(STATUS_LED, LOW);
digitalWrite(STATUS_LED, 0);
digitalWrite(STATUS_LED, 123);
digitalWrite(STATUS_LED, 255);
```

Wheels
```
sparki.moveForward(distance_centimeters);
sparki.moveBackward(distance_centimeters);
sparki.moveLeft(angle_degrees);
sparki.moveRight(angle_degrees);
sparki.moveStop();
```

Servo
```
sparki.servo(angle_degrees);
```

Stepper Motors
```
sparki.motorRotate(int motor, int direction, int speed);
sparki.motorStop(int motor);
```

Ultrasonic Range Finder
```
sparki.ping();
```