# Day 2

LED LIGHT PROJECT WITH JAVA

What we need?

- Raspberry Pi
- 1 Resistor
- 2 Wires
- 1 LED Light
- Electrical Tape

STEPS

1) Connect the resistor to the longer leg of LED light by twisting. (Tape over the connection point if you have trouble)
2) Connect the green cable to the other end of resistor by twisting the metal parts of the cables. (Tape over the connection point if you have trouble)
3) Connect the yellow cable to the shorter leg of LED light by twisting (Tape over the connection point if you have trouble)
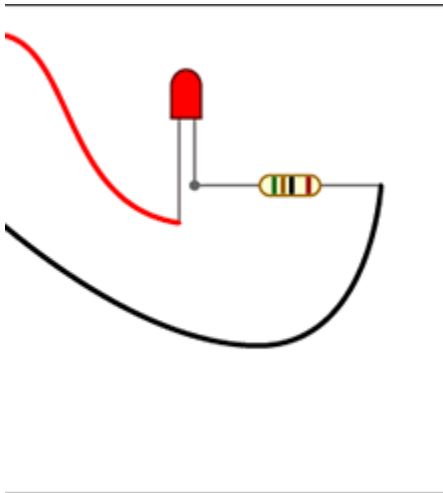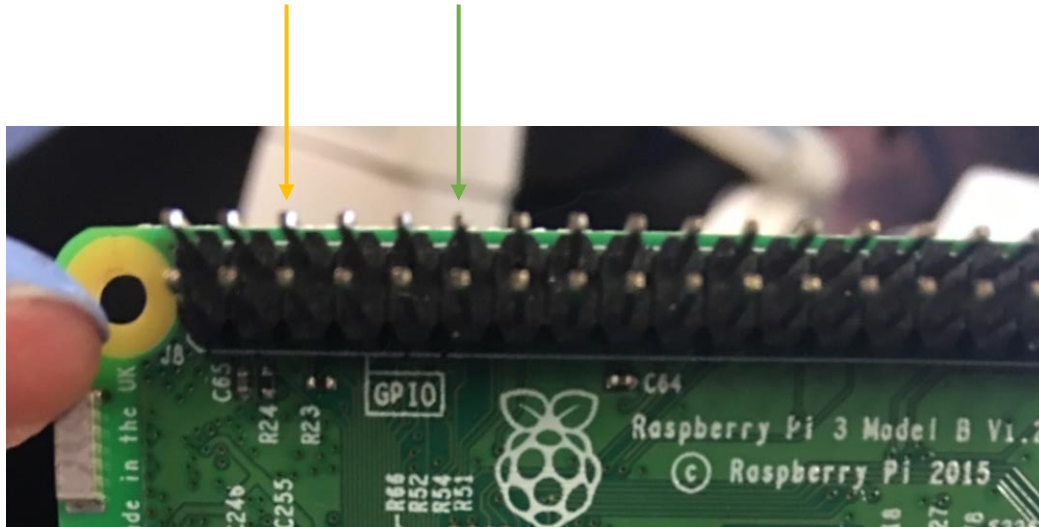

4) Take of your raspberry pi out the boxes. Build all the connections again.

5) Connect your long cable to the upper left pin on your Raspberry PI. And your short yellow cable to the 3rd pin as in below schema.

6) See if the light is on. If not, check your connections, tape them again.

7) If you passed the 6th step, connect your green cable to 6th pin.

8) Open your BlueJ Editor. Create a new Project with a name, that you pick.
9) Click Porject→ Import.
10) Navigate to your Desktop and select PI4J folder. Then, click to import.
11) Create a new class names LedLight.
12) Write the code below to the LedLight class

```java
import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;

/**
 * This example code demonstrates how to perform simple state
 * control of a GPIO pin on the Raspberry Pi.
 *
 * @author Robert Savage
 */
public class ControlGpioExample {

    public static void main(String[] args) throws InterruptedException {

        System.out.println("<--Pi4J--> GPIO Control Example ... started.");

        // create gpio controller
        final GpioController gpio = GpioFactory.getInstance();

        // provision gpio pin #01 as an output pin and turn on
```

```java
        final GpioPinDigitalOutput pin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01,
    "MyLED", PinState.HIGH);

        // set shutdown state for this pin
        pin.setShutdownOptions(true, PinState.LOW);

        System.out.println("--> GPIO state should be: ON");

        Thread.sleep(5000);

        // turn off gpio pin #01
        pin.low();
        System.out.println("--> GPIO state should be: OFF");

        Thread.sleep(5000);

        // toggle the current state of gpio pin #01 (should turn on)
        pin.toggle();
        System.out.println("--> GPIO state should be: ON");

        Thread.sleep(5000);

        // toggle the current state of gpio pin #01  (should turn off)
        pin.toggle();
        System.out.println("--> GPIO state should be: OFF");

        Thread.sleep(5000);

        // turn on gpio pin #01 for 1 second and then off
        System.out.println("--> GPIO state should be: ON for only 1 second");
        pin.pulse(1000, true); // set second argument to 'true' use a blocking call

        // stop all GPIO activity/threads by shutting down the GPIO controller
        // (this method will forcefully shutdown all GPIO monitoring threads and scheduled tasks)
        gpio.shutdown();

        System.out.println("Exiting ControlGpioExample");
    }
}
```

13) Compile your class
14) Right click on your class and run the main method

Your light will turn on. Then, turn off for 2 seconds and then turn on again.