DEVOXX™

Raspberry Pi with Java 8 + Pi4J

Robert Savage
Software Architect, MTS
Harman International

What is Pi4J?

Pi4J is an open-source project providing a library for Java programmers to interact with the low-level I/O capabilities on the Raspberry Pi platform.

- Open Source Project
- Low Level I/O Library
- Object-Oriented API
- Event Based
- Java & C (JNI + Native)

www.pi4j.com

Pi4J Supported I/O

Digital Interfaces

- GPIO (General Purpose Input/Output)

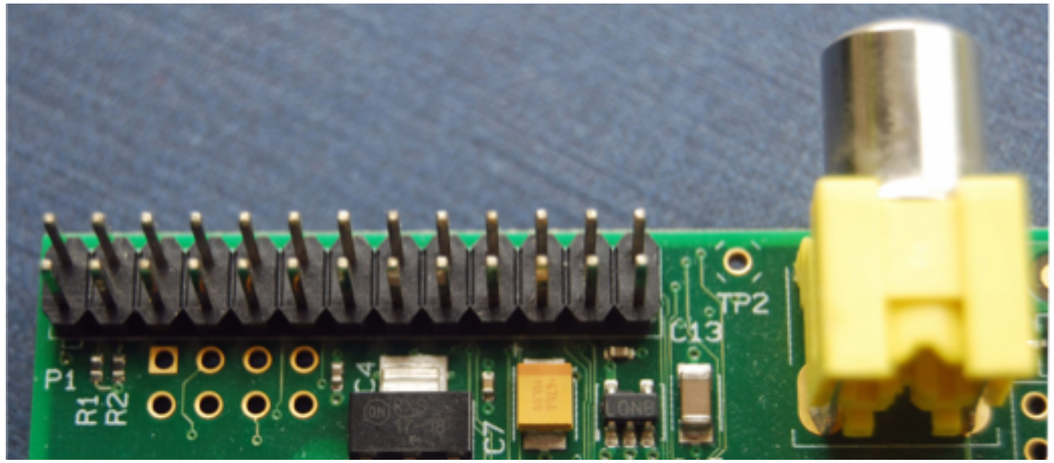
Data Interfaces

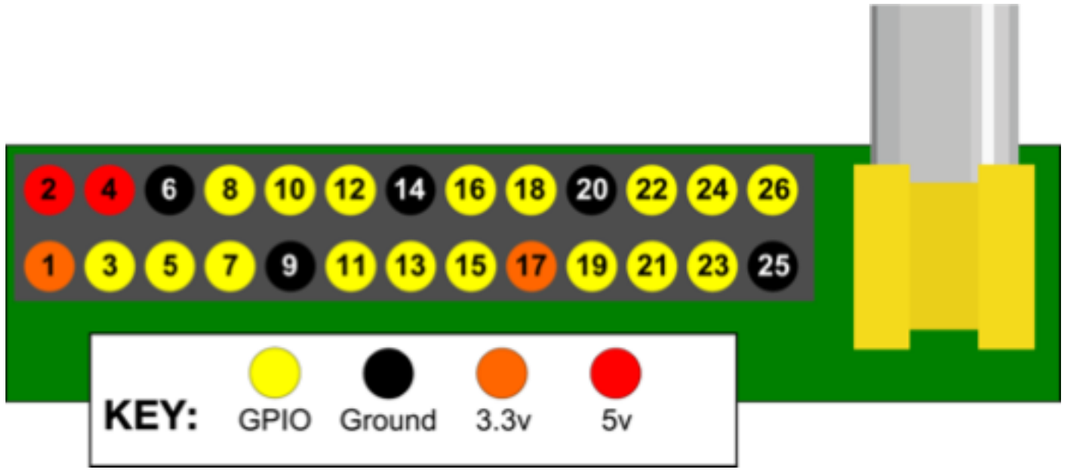
- UART, SERIAL (Universal Asynchronous Receiver/Transmitter)
- SPI (Serial Peripheral Interface)
- I^2C (Inter-Integrated Circuit)

Analog Interfaces*

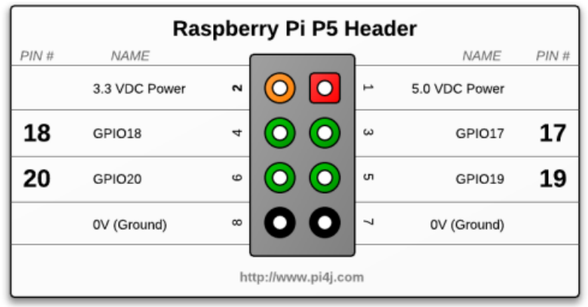
GPIO

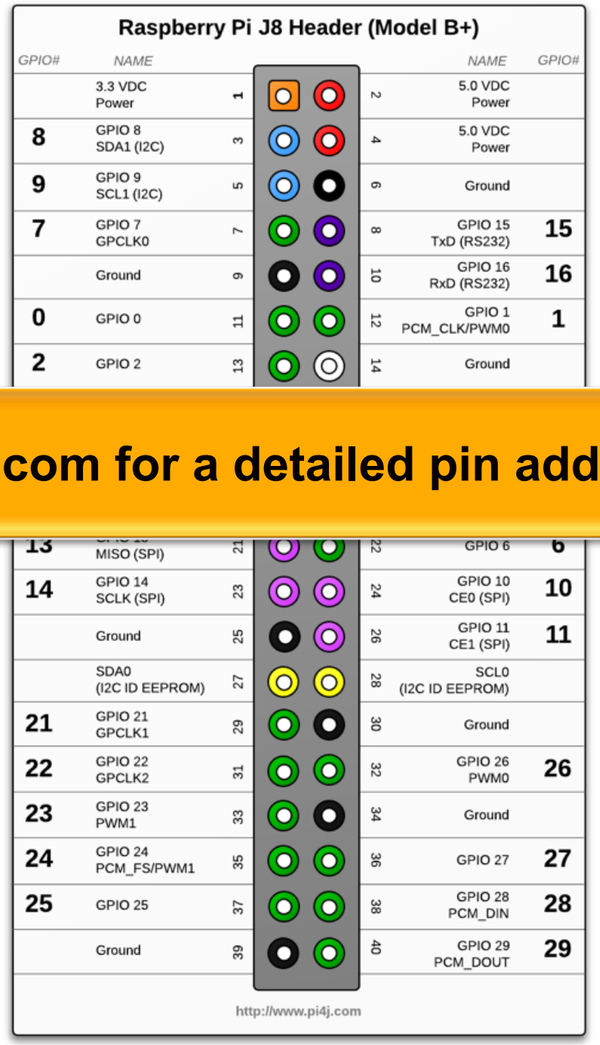
- **Input or Output**
- **Digital States**
  - HIGH = +3.3 VDC
  - LOW = 0 VDC
- **RPi Models**
  - A & B = ~21 GPIO
  - B+ = 28 GPIO
  - Compute Module = 46 GPIO

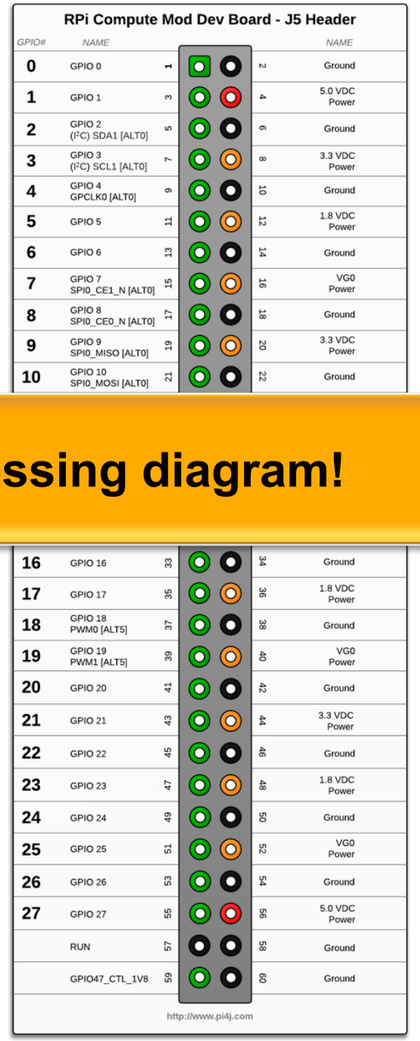


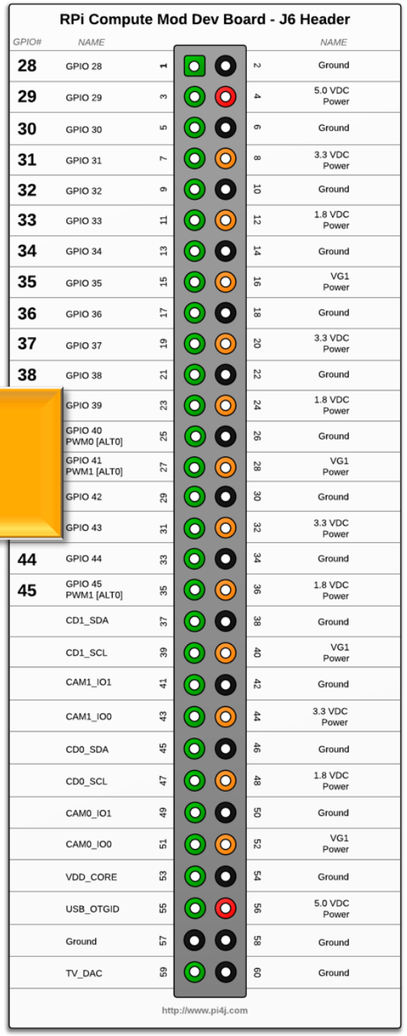


Pi4J GPIO Pin Addressing





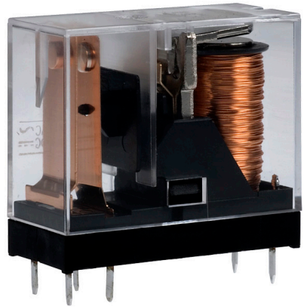




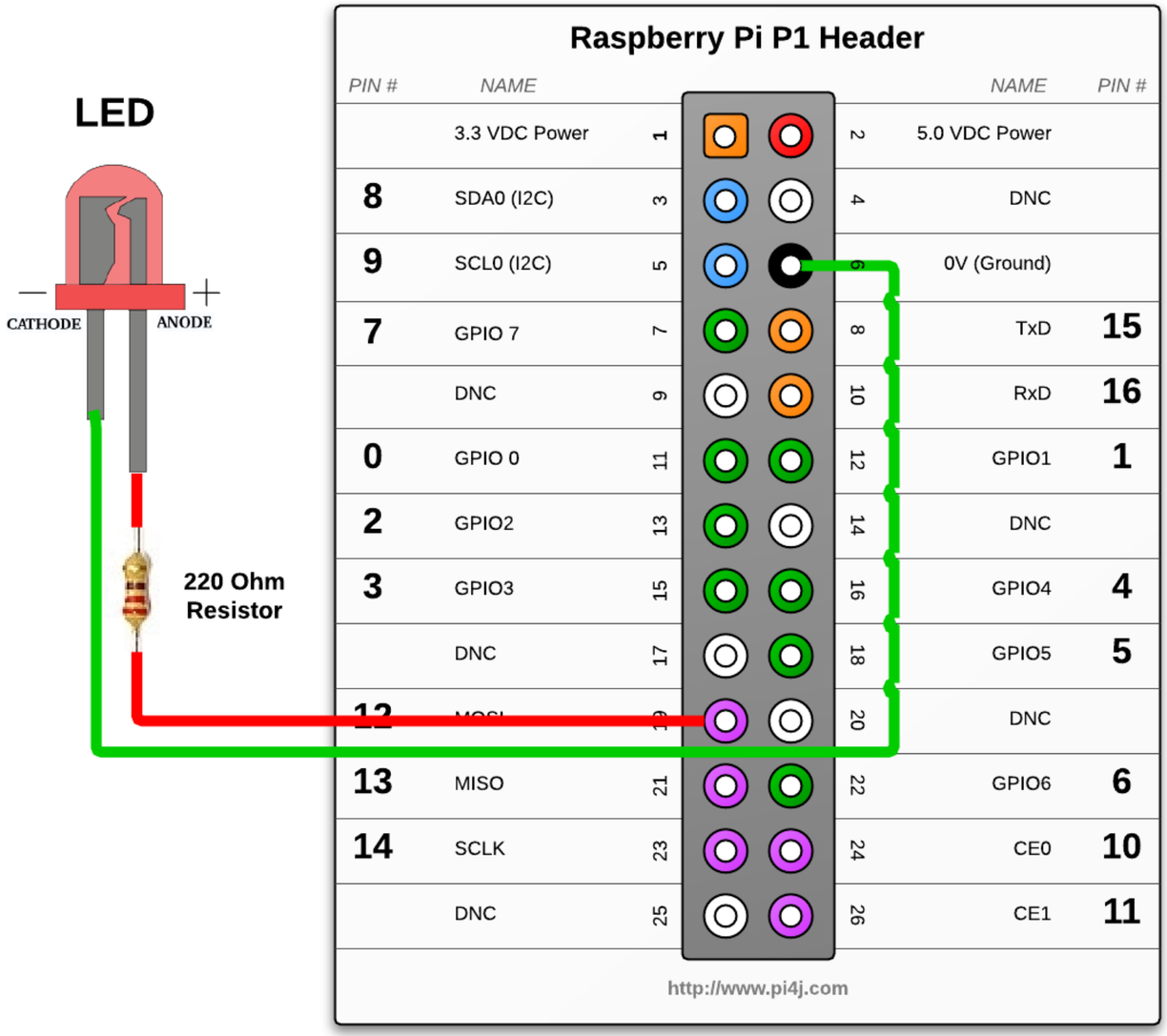
Visit pi4.com for a detailed pin addressing diagram!

GPIO Outputs

- Control Things (ON & OFF)



GPIO Output Circuit



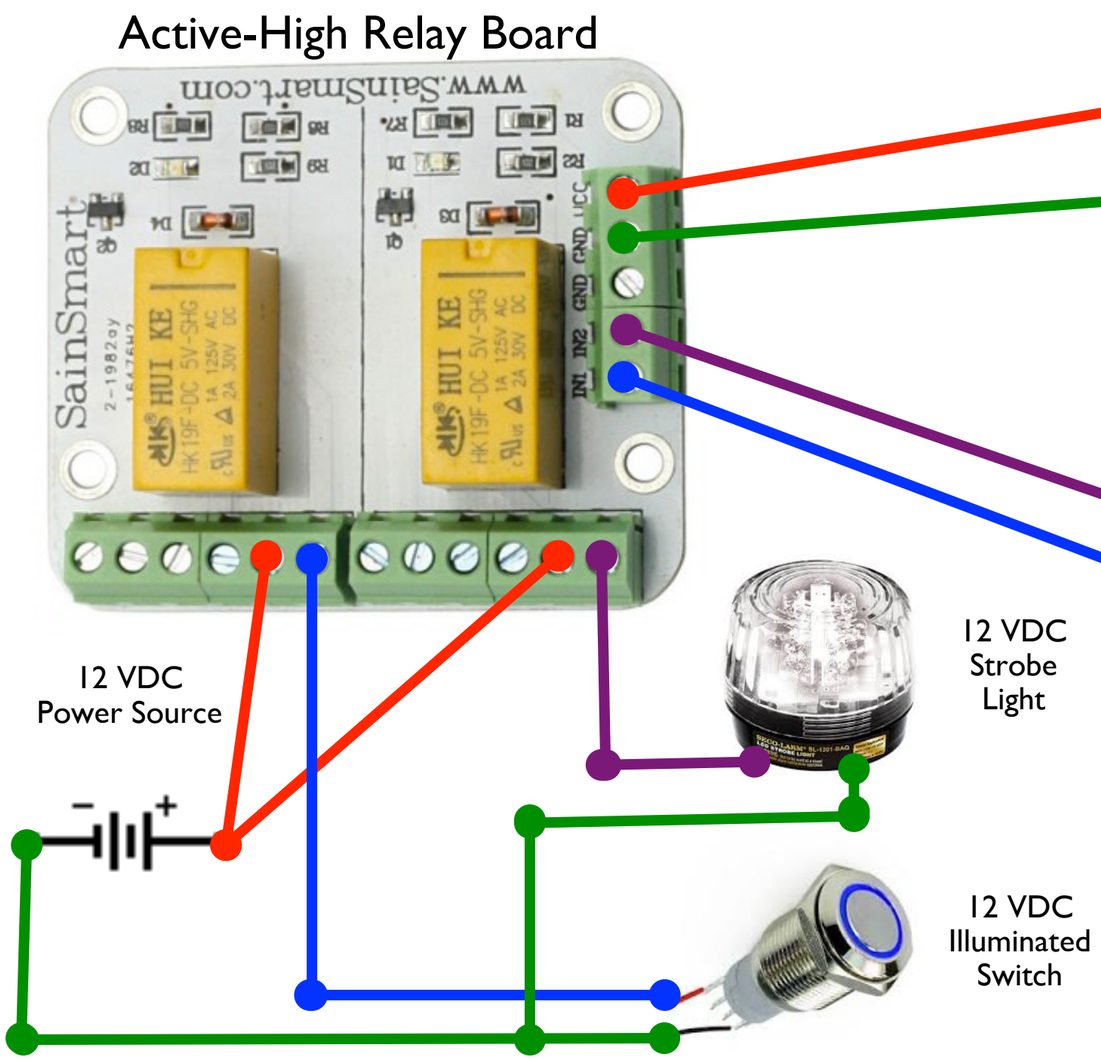
GPIO Output Example

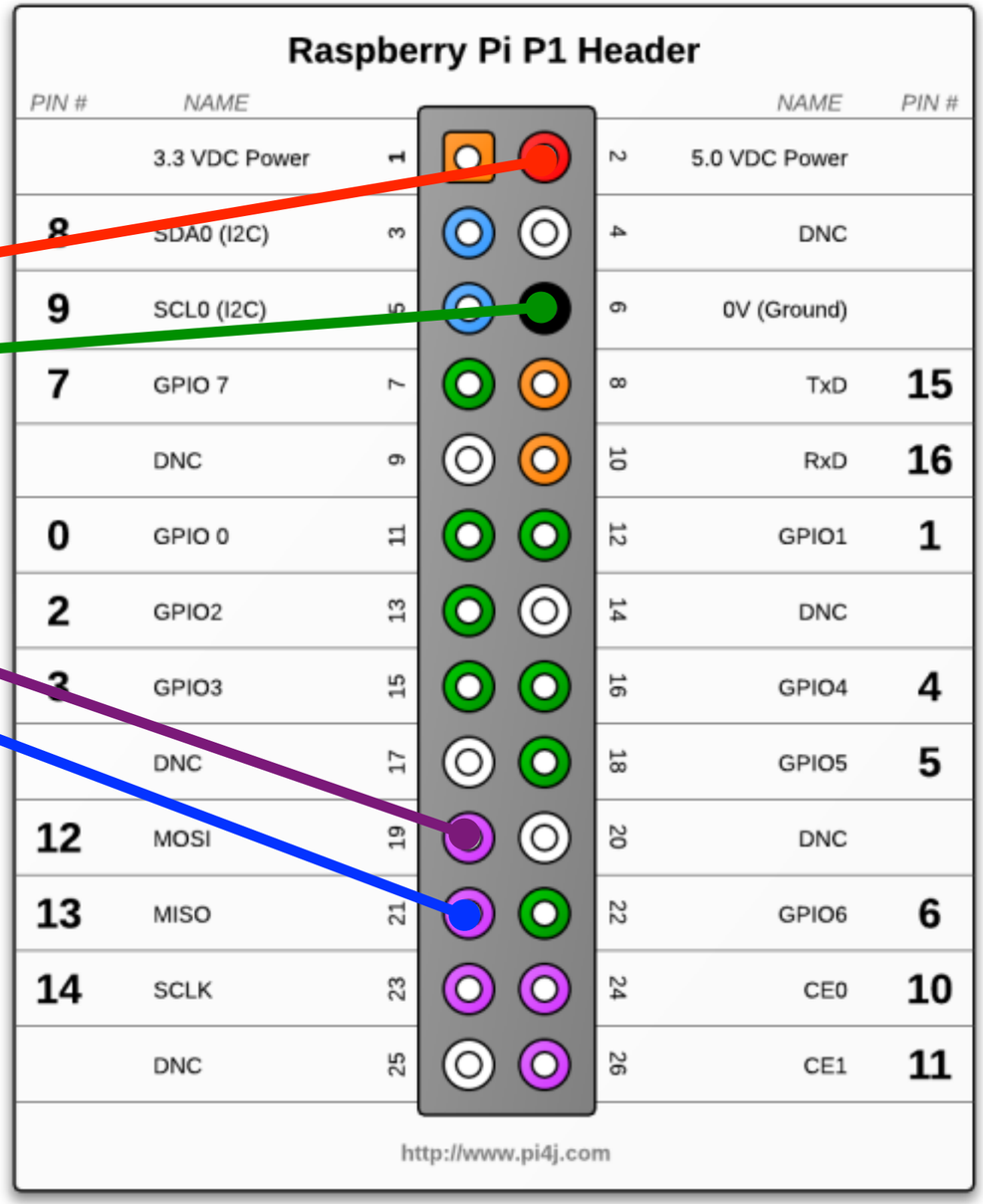
```
// create GPIO controller
final GpioController gpio = GpioFactory.getInstance();

// create GPIO output pin
final GpioPinDigitalOutput output =
                gpio.provisionDigitalOutputPin(
                        RaspiPin.GPIO_12, PinState.LOW);

// control GPIO output pin
output.high();
output.low();
output.toggle();        // invert current state
output.pulse(1000);     // set state for a limited duration
```

GPIO Output Circuit



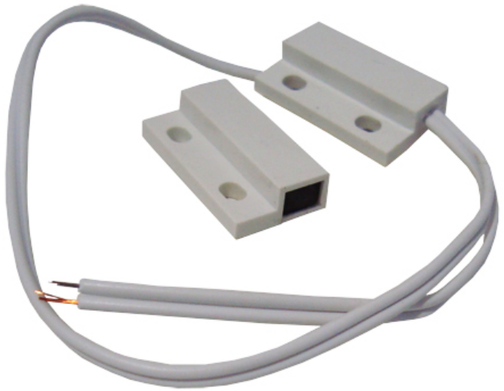


GPIO Output Demo

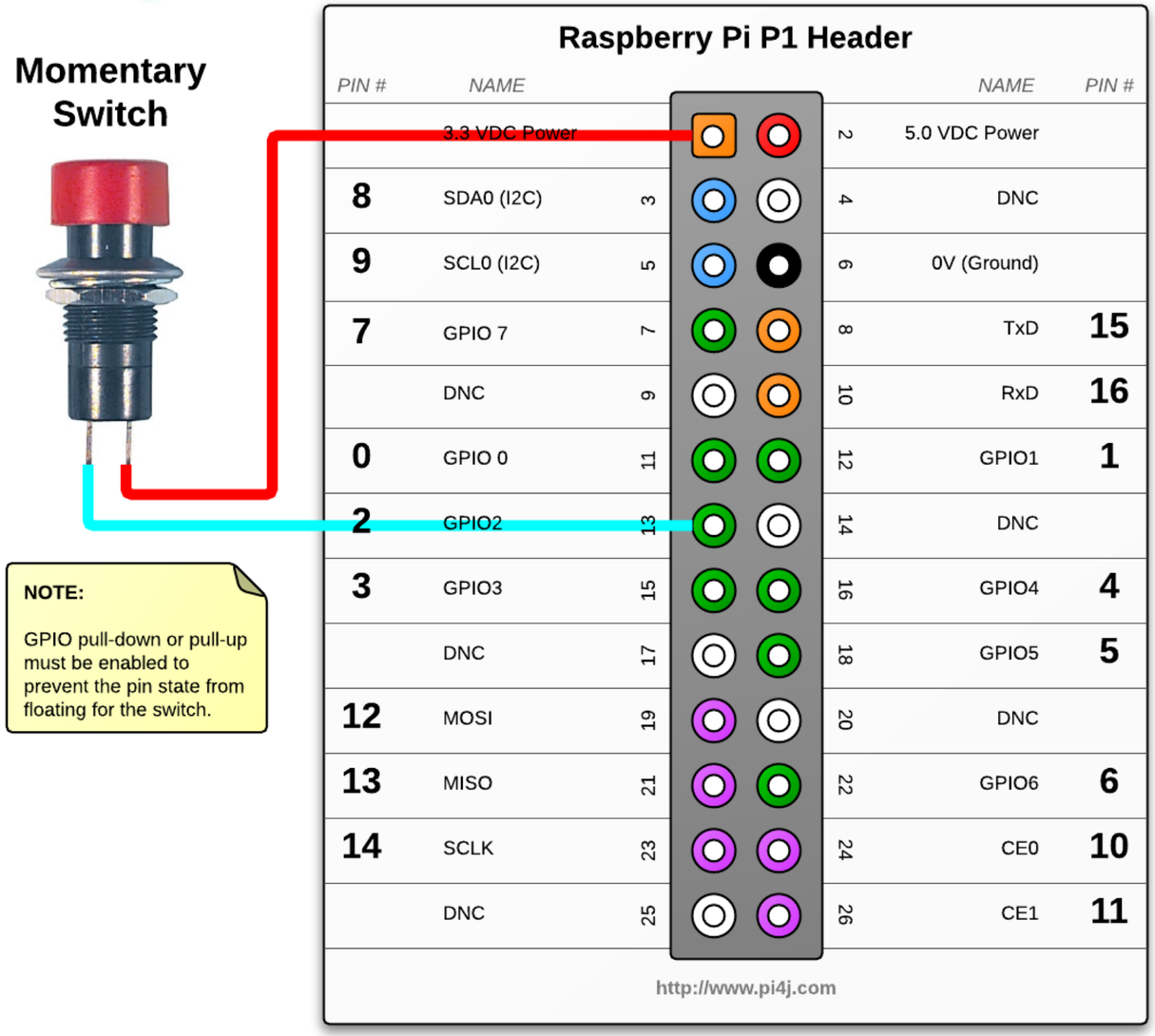
- Sample Code
- Live Demo

GPIO Inputs

- Monitor Things
  - ON & OFF
  - Open & Close



GPIO Input Circuit

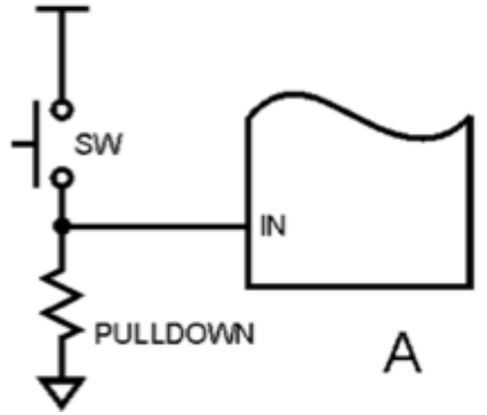


GPIO Input Reference

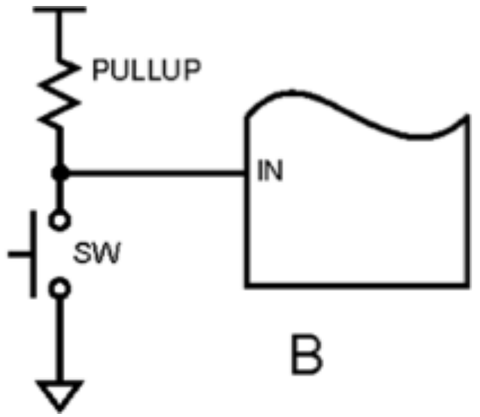
- GPIO inputs require a "*reference*" voltage.
- Without a reference, a GPIO pin can "*float*"
- The Raspberry Pi includes internal ***PULL-UP*** and ***PULL-DOWN*** resistor settings that can be configured via Pi4J.

GPIO Input Reference

- **PULL-DOWN**
  Resistance provides a reference (bias) to GROUND (0 VDC). If your circuit expects to provide +3.3VDC to signal the GPIO pin HIGH, then you need a PULL-DOWN reference.



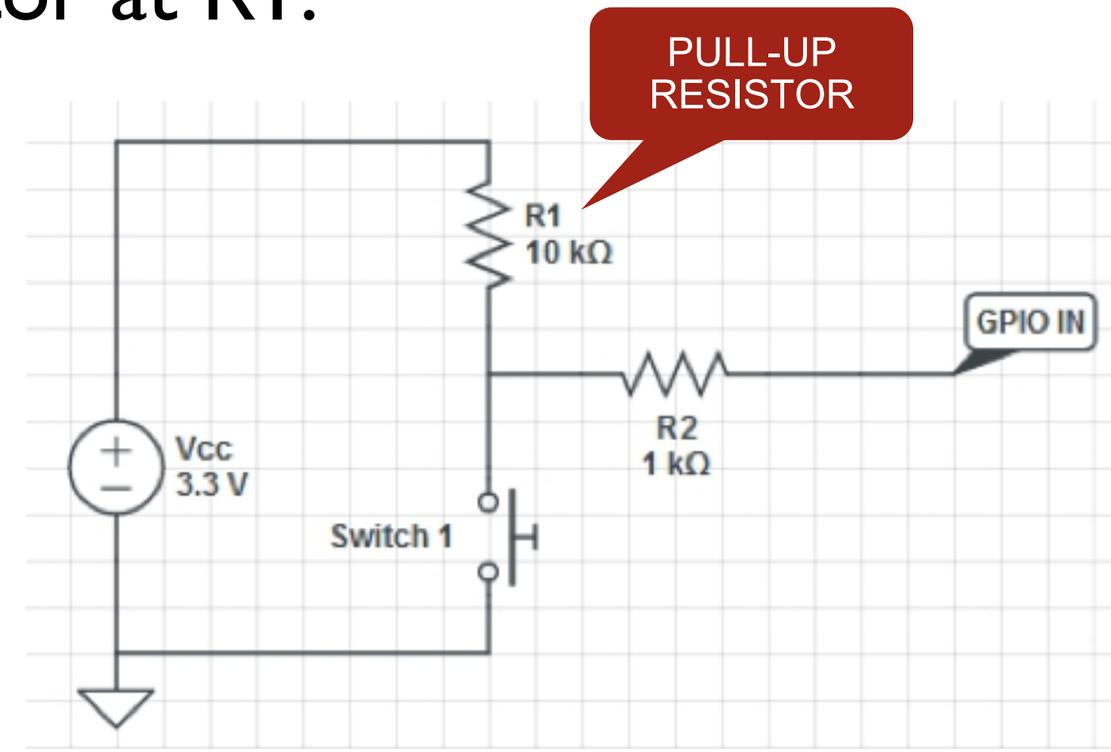
- **PULL-UP**
  Resistance provides a reference (bias) to +3.3 VDC. If your circuit expects to provide GROUND to signal the GPIO pin LOW, then you need a PULL-UP reference.



GPIO Input Reference

Alternatively, you can build the PULL-UP or PULL-DOWN reference in the hardware circuit. The circuit below demonstrates a PULL-UP resistor at R1.

This circuit signals the GPIO input pin to LOW when the switch closes the circuit and a path to GROUND is complete.



GPIO Input Example

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

// create a GPIO input pin
final GpioPinDigitalInput input = gpio.provisionDigitalInputPin(
                                    RaspiPin.GPIO_02,
                                    PinPullResistance.PULL_DOWN);
```

GPIO Input Listener Example

Java 8 Lambda

```
// create event listener for GPIO input pin
input.addListener((GpioPinListenerDigital)
                        (GpioPinDigitalStateChangeEvent event) -> {

    // set output state to match the input state
    output.setState(event.getState());
});
```

GPIO Input Demo

- Sample Code
- Live Demo

Pi4J Component API

- The component APIs provides an abstraction layer from the hardware I/O layer.
- This allows hardware design/circuitry to change with *less* impact to your implementation code.
- For example, a RELAY could be controlled from GPIO, RS232, SPI, or I2C. You program defines the RELAY impl up front based on the hardware interface, but the rest of your program logic works against the RELAY component interface and not the direct hardware /communication IO interfaces.

Component Interfaces

- Keypad
- Light / LED
- Dimmable Light
- LCD
- Power Controller
- Relay
- Momentary Switch
- Toggle Switch
- Analog Sensor
- Distance Sensor
- Motion Sensor
- Temperature Sensor

GPIO Components Example

```
// create LED component
final Light light = new GpioLightComponment(output);

// usage example
light.on(); (or) light.off();


// create momentary switch component
final MomentarySwitch ms = new GpioMomentarySwitchComponment(
                    input,
                    PinState.LOW,    // “OFF” pin state
                    PinState.HIGH); // “ON” pin state
```

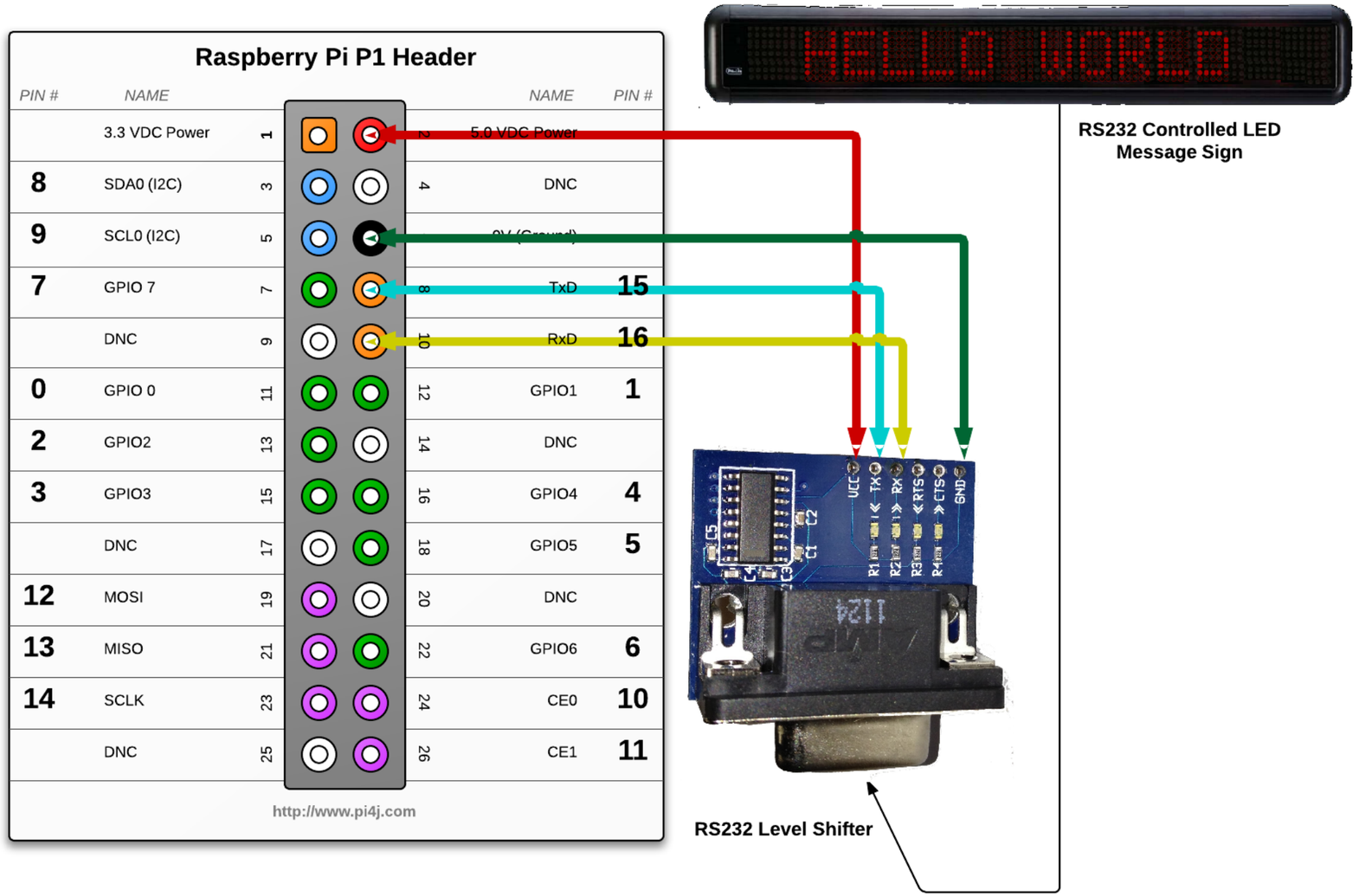
Component Demo

- Sample Code
- Live Demo

UART / Serial / RS-232

- The Raspberry Pi supports on on-board UART for serial communication.
- Pi4J supports basic serial communication.
- To use RS232 serial communication a level-shifter is required to convert between the TTL voltage levels (+3.3 VDC) and those required for RS232 communication.

RS-232 Level Shifter



USB to RS-232 Adapter

In addition to the on-board UART, you can also use a USB to RS232 adapter connected to the Raspberry Pi to provide RS232 serial communication.

UART / RS-232 Example

```
// create an instance of the serial communications class
final Serial serial = SerialFactory.createInstance();

// open the default serial port provided on the P1 header
// (this is where our LED reader is connected)
serial.open(Serial.DEFAULT_COM_PORT, 2400);

// send "Hello World" message to sign
serial.writeln("<ID01><PA><CL>Hello World!  --  ");
serial.writeln("<ID01><RPA>");
```

UART / RS-232 Listener Example

Java 8
Lambda

```
// create and register the serial data listener
serial.addListener((SerialDataListener) (SerialDataEvent event) -> {

        // print out the data received to the console
        System.out.println(event.getData());

});
```

UART / RS-232 Demo

- Sample Code
- Live Demo

Putting It All Together

Lets create a real-world demo using a Raspberry Pi, Java, Pi4J, GPIO Inputs & Outputs, and RS232 (Serial) devices to do something *useful*.

Demo Project Goal

- Create a sophisticated model rocket launching platform.

Demo Project Requirements

- Implement a safety key switch to arm the rocket and protect from accidental launch ignition.

- Implement a launch activation button to initiate a launch sequence.

Demo Project Requirements

- Implement an audible safety alarm to alert spectators of launch ready conditions.

- Implement a visible alert device to notify spectators of launch ready conditions.

Demo Project Requirements

- Implement an abort button to abort the launch sequence.

- Implement a safety IR beam detector to abort launch if a person approaches the launch pad.

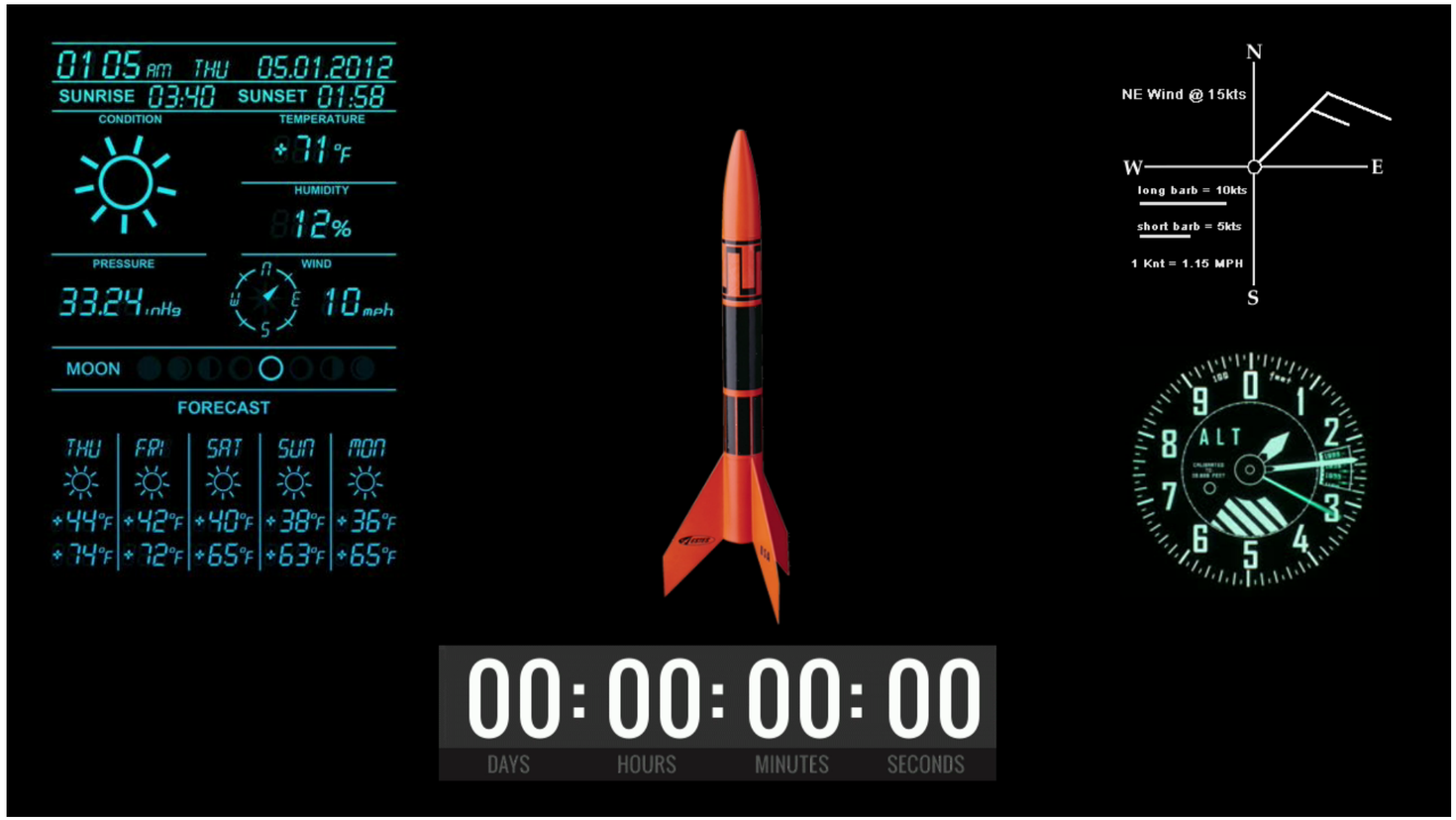
Demo Project Requirements

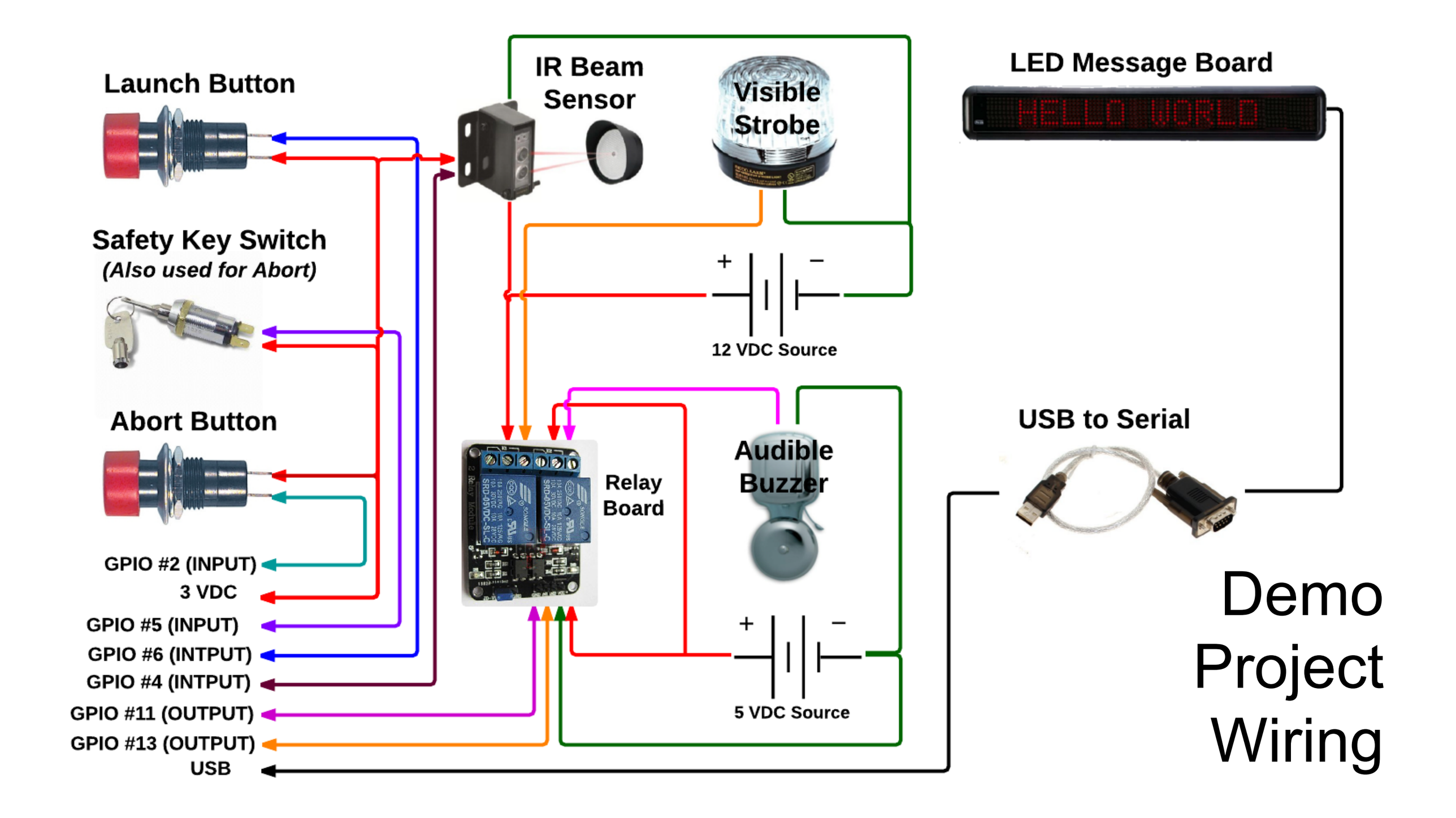
- Implement an LED message board to show spectators the countdown and phases of the launch.

- Implement a countdown timer for the launch sequence.

Demo Project Requirements

- Implement an on-screen console/dashboard




Launch Button
IR Beam Sensor
Visible Strobe
LED Message Board
HELLO WORLD
Safety Key Switch
(Also used for Abort)
+
−
12 VDC Source
Abort Button
USB to Serial
Relay Board
Audible Buzzer
GPIO #2 (INPUT)
3 VDC
GPIO #5 (INPUT)
GPIO #6 (INTPUT)
GPIO #4 (INTPUT)
GPIO #11 (OUTPUT)
GPIO #13 (OUTPUT)
USB
5 VDC Source
Demo Project Wiring

Demo Project

- Sample Code
- Live Demo

Savage Home Automation Blog

savagehomeautomation.com

@savageautomate

The Pi4J Project

pi4j.com

@pi4j