

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

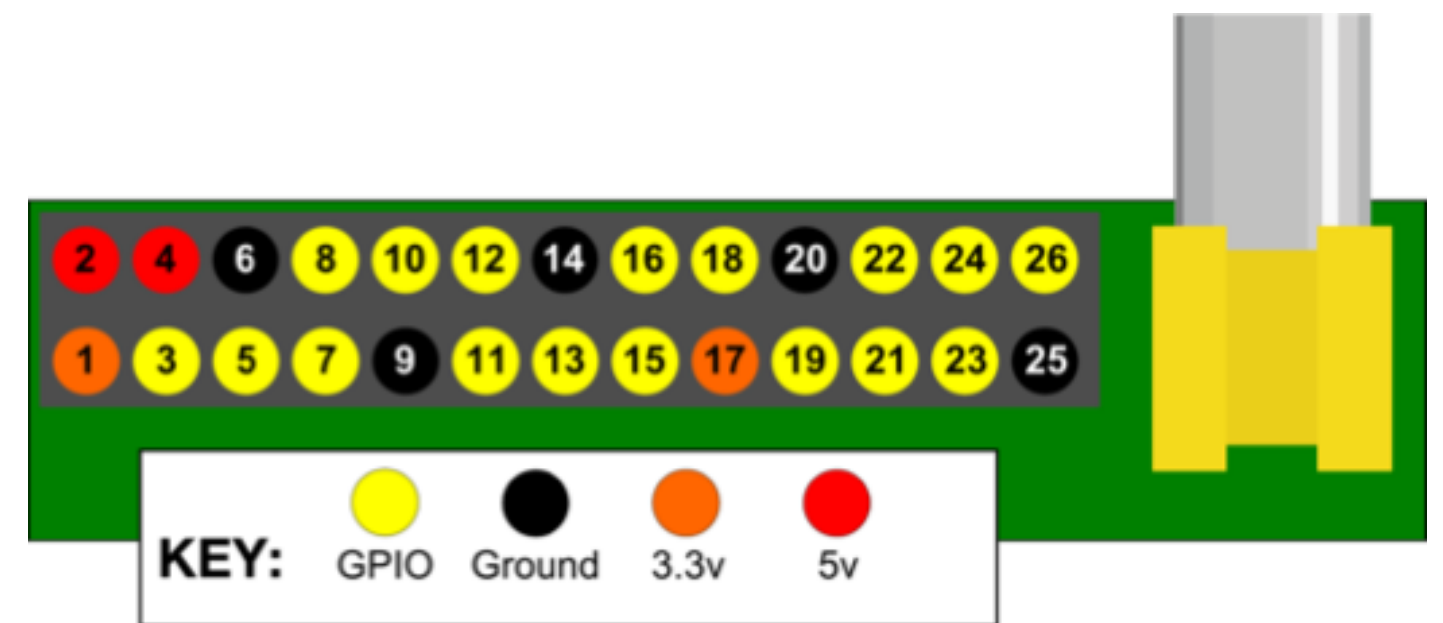
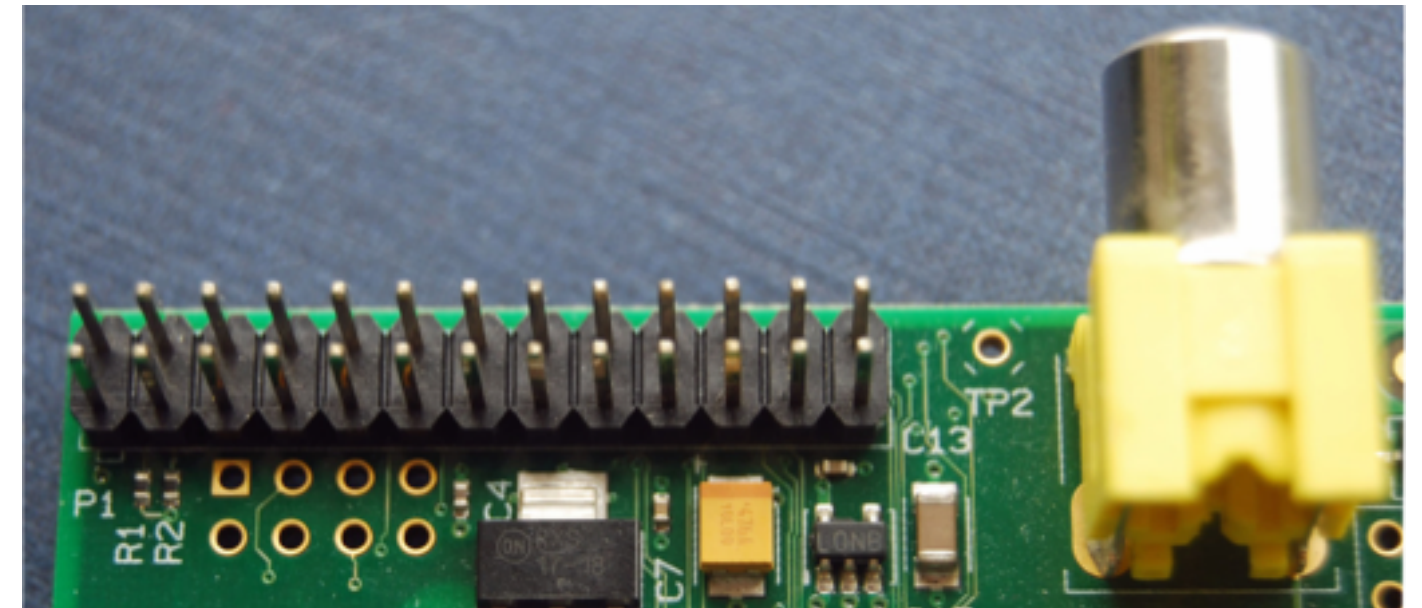
- **UART, SERIAL** (Universal Asynchronous Receiver/Transmitter)
- **SPI** (Serial Peripheral Interface)
- **I²C** (Inter-Integrated Circuit)

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

Raspberry Pi P1 Header			
PIN #	NAME		NAME PIN #
	3.3 VDC Power	1	5.0 VDC Power 2
8	SDA0 (I2C)	3	DNC 4
9	SCL0 (I2C)	5	0V (Ground) 6
7	GPIO 7	7	TxD 15
	DNC	9	RxD 16
0	GPIO 0	11	GPIO1 1
2	GPIO2	13	DNC 14
3	GPIO3	15	
	DNC	17	
12	MOSI	19	
13	MISO	21	GPIO6 6
14	SCLK	23	CE0 10
	DNC	25	CE1 11

<http://www.pi4j.com>

Raspberry Pi J8 Header (Model B+)			
GPIO#	NAME		NAME GPIO#
	3.3 VDC Power	1	5.0 VDC Power 2
8	GPIO 8 SDA1 (I2C)	3	5.0 VDC Power 4
9	GPIO 9 SCL1 (I2C)	5	Ground 6
7	GPIO 7 GPCLK0	7	GPIO 15 TxD (RS232) 15
	Ground	9	GPIO 16 RxD (RS232) 16
0	GPIO 0	11	GPIO 1 PCM_CLK/PWM0 1
2	GPIO 2	13	Ground 14
13	MISO (SPI)	12	GPIO 6 6
14	GPIO 14 SCLK (SPI)	23	GPIO 10 CE0 (SPI) 10
	Ground	25	GPIO 11 CE1 (SPI) 11
	SDA0 (I2C ID EEPROM)	27	SCL0 (I2C ID EEPROM) 28
21	GPIO 21 GPCLK1	29	Ground 30
22	GPIO 22 GPCLK2	31	GPIO 26 PWM0 26
23	GPIO 23 PWM1	33	Ground 34
24	GPIO 24 PCM_FS/PWM1	35	GPIO 27 27
25	GPIO 25	37	GPIO 28 PCM_DIN 28
	Ground	39	GPIO 29 PCM_DOUT 29

<http://www.pi4j.com>

RPi Compute Mod Dev Board - J5 Header			
GPIO#	NAME		NAME
0	GPIO 0	1	Ground 2
1	GPIO 1	3	5.0 VDC Power 4
2	GPIO 2 (iFC) SDA1 [ALT0]	5	Ground 6
3	GPIO 3 (iFC) SCL1 [ALT0]	7	3.3 VDC Power 8
4	GPIO 4 GPCLK0 [ALT0]	9	Ground 10
5	GPIO 5	11	1.8 VDC Power 12
6	GPIO 6	13	Ground 14
7	GPIO 7 SPI0_CE1_N [ALT0]	15	VG0 Power 16
8	GPIO 8 SPI0_CE0_N [ALT0]	17	Ground 18
9	GPIO 9 SPI0_MISO [ALT0]	19	3.3 VDC Power 20
10	GPIO 10 SPI0_MOSI [ALT0]	21	Ground 22
16	GPIO 16	33	Ground 34
17	GPIO 17	35	1.8 VDC Power 36
18	GPIO 18 PWM0 [ALT5]	37	Ground 38
19	GPIO 19 PWM1 [ALT5]	39	VG0 Power 40
20	GPIO 20	41	Ground 42
21	GPIO 21	43	3.3 VDC Power 44
22	GPIO 22	45	Ground 46
23	GPIO 23	47	1.8 VDC Power 48
24	GPIO 24	49	Ground 50
25	GPIO 25	51	VG0 Power 52
26	GPIO 26	53	Ground 54
27	GPIO 27	55	5.0 VDC Power 56
	RUN	57	Ground 58
	GPIO47_CTL_1V8	59	Ground 60

<http://www.pi4j.com>

RPi Compute Mod Dev Board - J6 Header			
GPIO#	NAME		NAME
28	GPIO 28	1	Ground 2
29	GPIO 29	3	5.0 VDC Power 4
30	GPIO 30	5	Ground 6
31	GPIO 31	7	3.3 VDC Power 8
32	GPIO 32	9	Ground 10
33	GPIO 33	11	1.8 VDC Power 12
34	GPIO 34	13	Ground 14
35	GPIO 35	15	VG1 Power 16
36	GPIO 36	17	Ground 18
37	GPIO 37	19	3.3 VDC Power 20
38	GPIO 38	21	Ground 22
	GPIO 39	23	1.8 VDC Power 24
	GPIO 40 PWM0 [ALT0]	25	Ground 26
	GPIO 41 PWM1 [ALT0]	27	VG1 Power 28
	GPIO 42	29	Ground 30
	GPIO 43	31	3.3 VDC Power 32
44	GPIO 44	33	Ground 34
45	GPIO 45 PWM1 [ALT0]	35	1.8 VDC Power 36
	CD1_SDA	37	Ground 38
	CD1_SCL	39	VG1 Power 40
	CAM1_IO1	41	Ground 42
	CAM1_IO0	43	3.3 VDC Power 44
	CD0_SDA	45	Ground 46
	CD0_SCL	47	1.8 VDC Power 48
	CAM0_IO1	49	Ground 50
	CAM0_IO0	51	VG1 Power 52
	VDD_CORE	53	Ground 54
	USB_OTGID	55	5.0 VDC Power 56
	Ground	57	Ground 58
	TV_DAC	59	Ground 60

<http://www.pi4j.com>

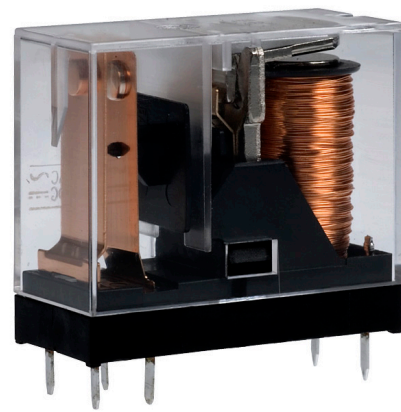
Visit pi4.com for a detailed pin addressing diagram!

Raspberry Pi P5 Header			
PIN #	NAME		NAME PIN #
	3.3 VDC Power	2	5.0 VDC Power 1
18	GPIO18	4	GPIO17 17
20	GPIO20	6	GPIO19 19
	0V (Ground)	8	0V (Ground) 7

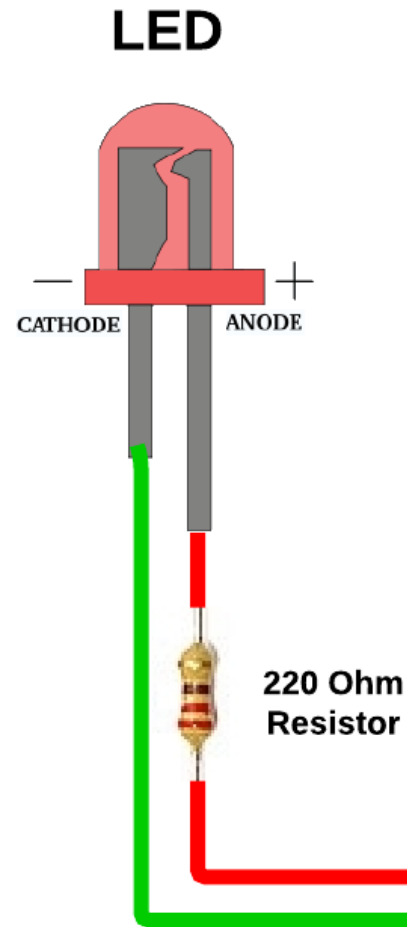
<http://www.pi4j.com>

GPIO Outputs

- Control Things (ON & OFF)



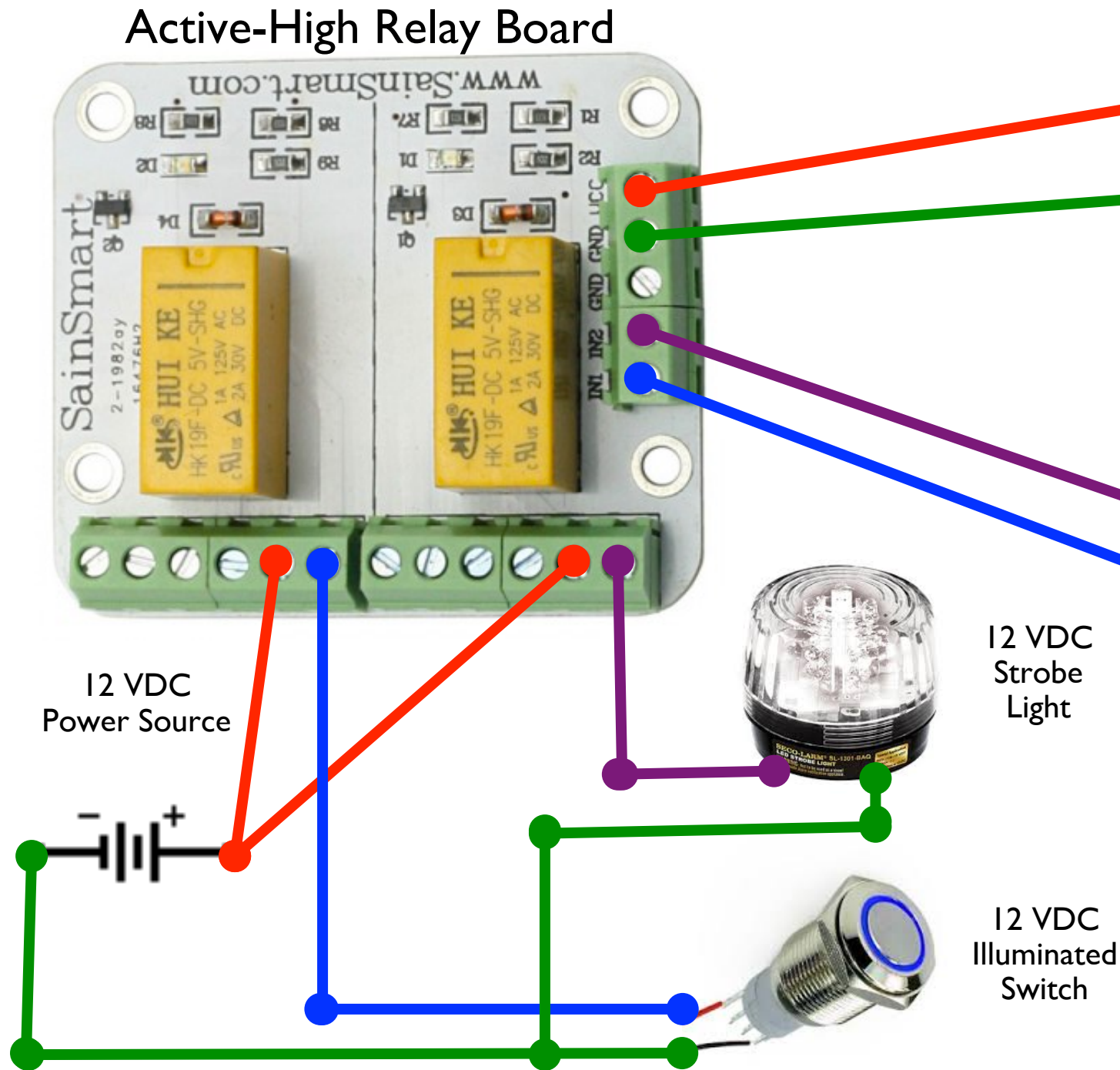
GPIO Output Circuit



Raspberry Pi P1 Header				
PIN #	NAME		NAME	PIN #
	3.3 VDC Power	1	5.0 VDC Power	2
8	SDA0 (I2C)	3	DNC	4
9	SCL0 (I2C)	5	0V (Ground)	6
7	GPIO 7	7	TxD	15
	DNC	9	RxD	16
0	GPIO 0	11	GPIO1	1
2	GPIO2	13	DNC	
3	GPIO3	15	GPIO4	4
	DNC	17	GPIO5	5
12	MOSI	19	DNC	
13	MISO	21	GPIO6	6
14	SCLK	23	CE0	10
	DNC	25	CE1	11

<http://www.pi4j.com>

GPIO Output Circuit



PIN #	NAME		NAME	PIN #
	3.3 VDC Power	1	5.0 VDC Power	2
8	SDA0 (I2C)	3	DNC	4
9	SCL0 (I2C)	5	0V (Ground)	6
7	GPIO 7	7	TxD	15
	DNC	9	RxD	16
0	GPIO 0	11	GPIO1	1
2	GPIO2	13	DNC	14
3	GPIO3	15	GPIO4	4
	DNC	17	GPIO5	5
12	MOSI	19	DNC	20
13	MISO	21	GPIO6	6
14	SCLK	23	CE0	10
	DNC	25	CE1	11

<http://www.pi4j.com>

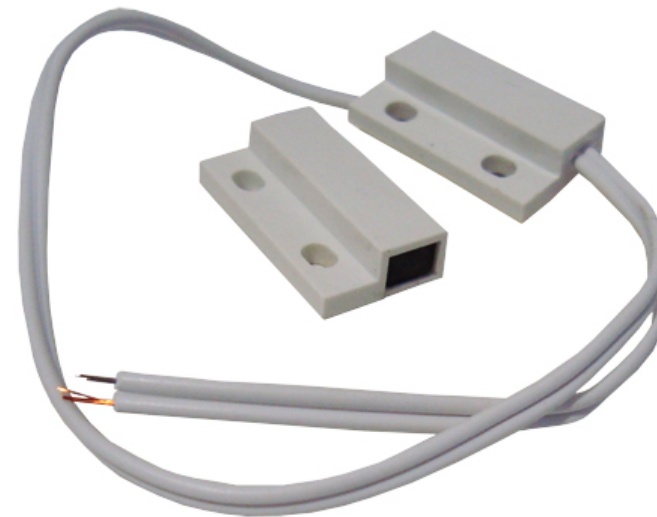


GPIO Output Demo

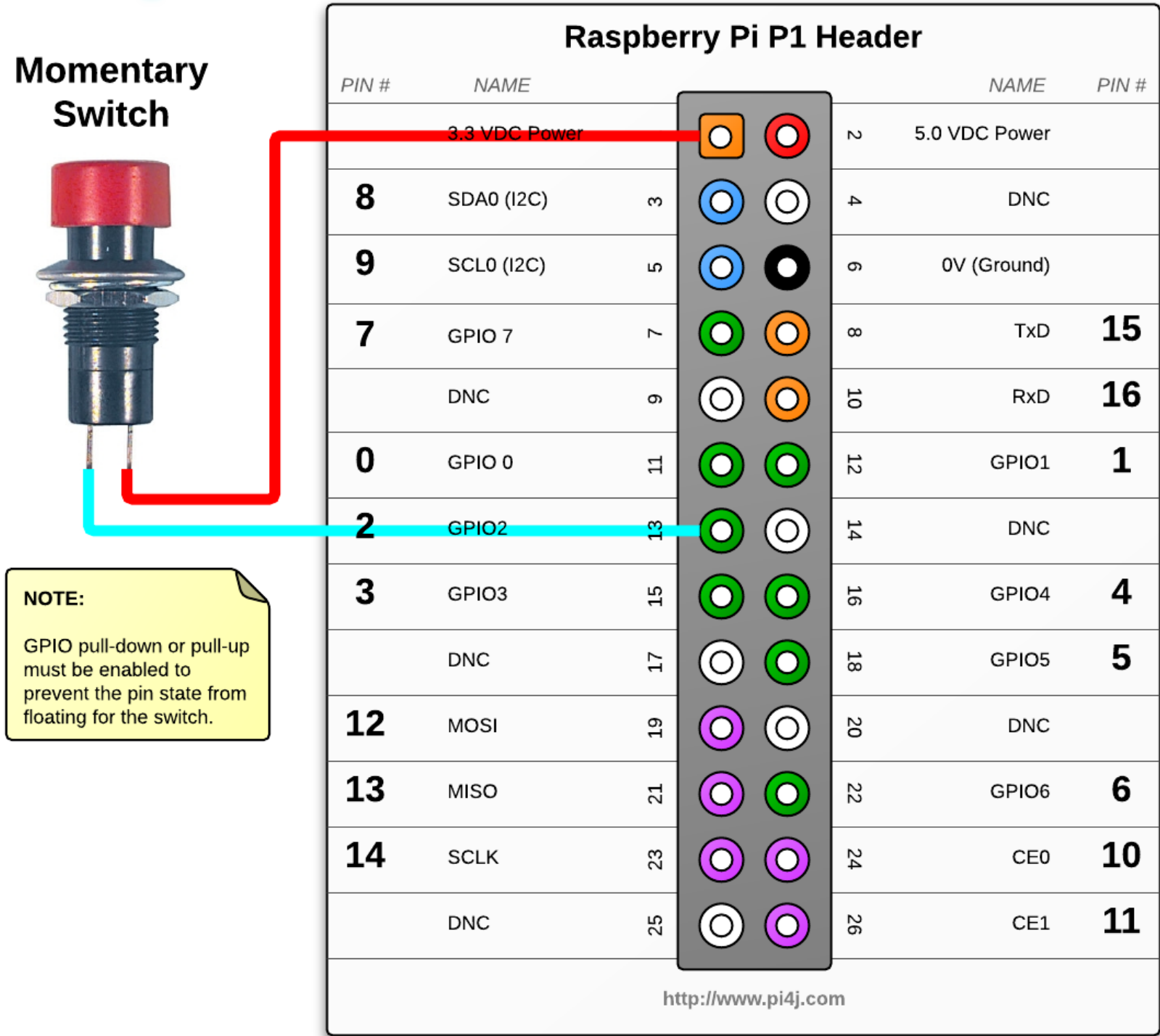
- Sample Code
- Live Demo

GPIO Inputs

- Monitor Things
 - ON & OFF
 - Open & Close



GPIO Input Circuit





GPIO Input Reference

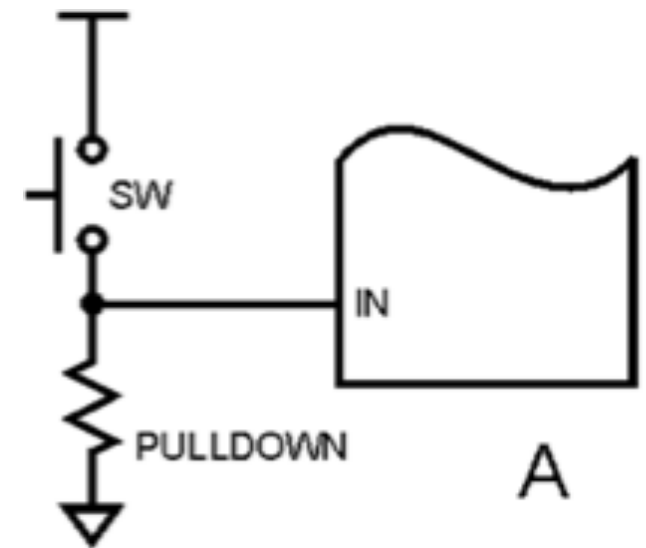
- GPIO inputs require a “*reference*” voltage.
- Without a reference, a GPIO pin can “*float*”
- The Raspberry Pi includes internal ***PULL-UP*** and ***PULL-DOWN*** resistor settings that can be configured via Pi4j.



GPIO Input Reference

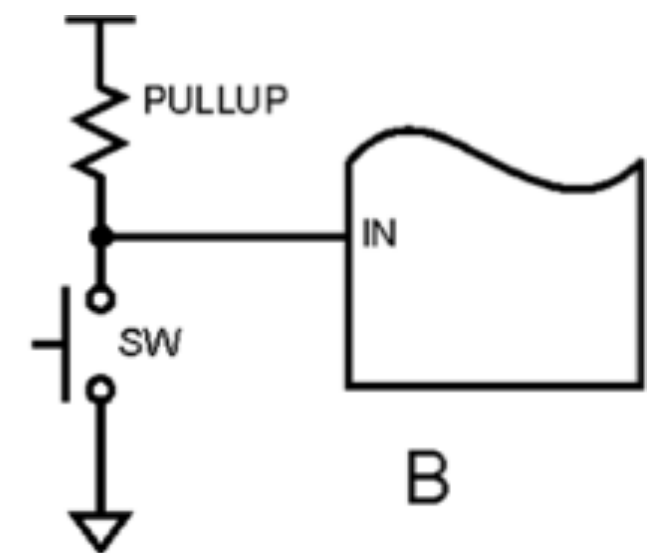
- **PULL-DOWN**

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.

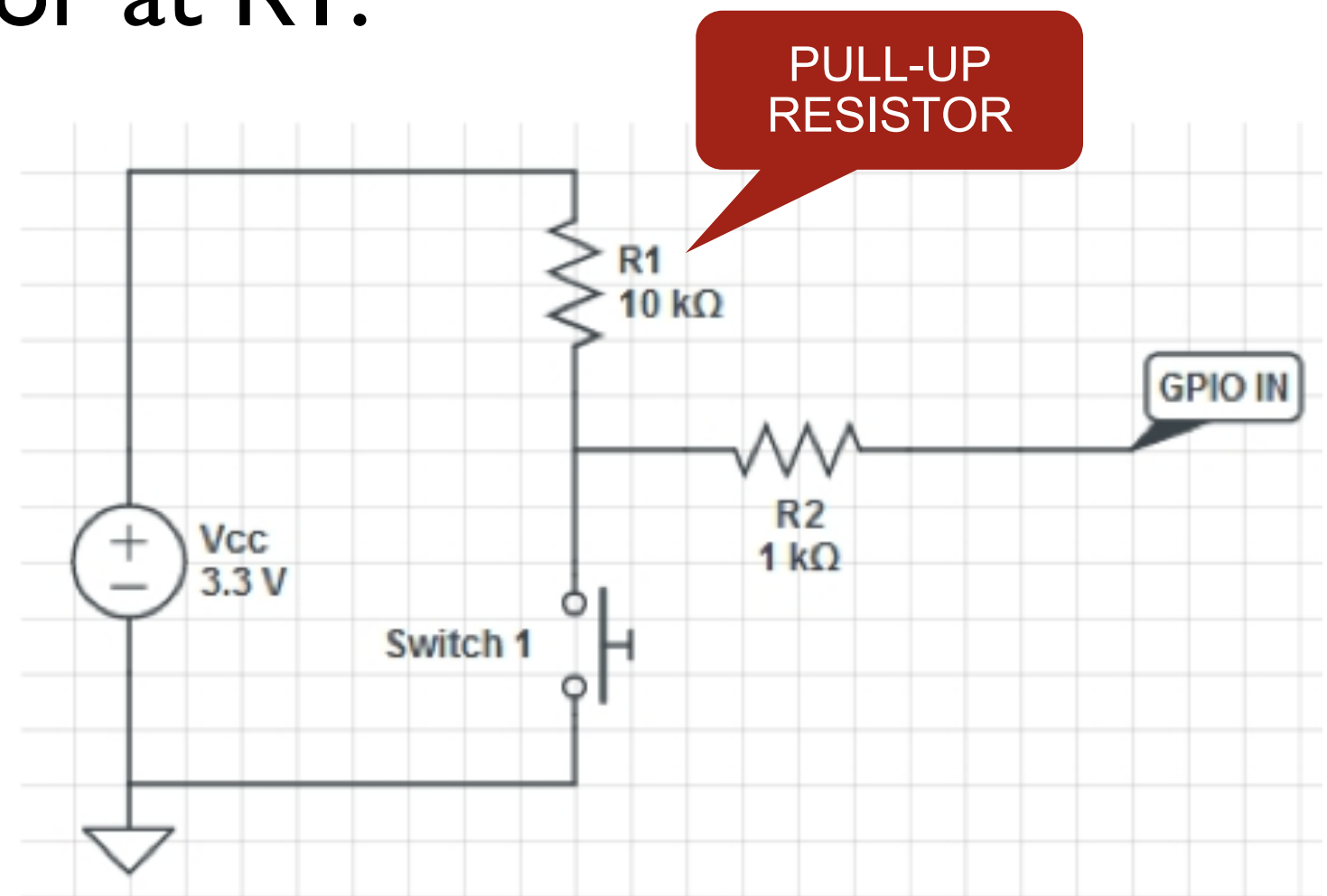




GPIO Input Reference

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. Your 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 GpioLightComponent(output);

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

// create momentary switch component
final MomentarySwitch ms = new GpioMomentarySwitchComponent(
    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-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.

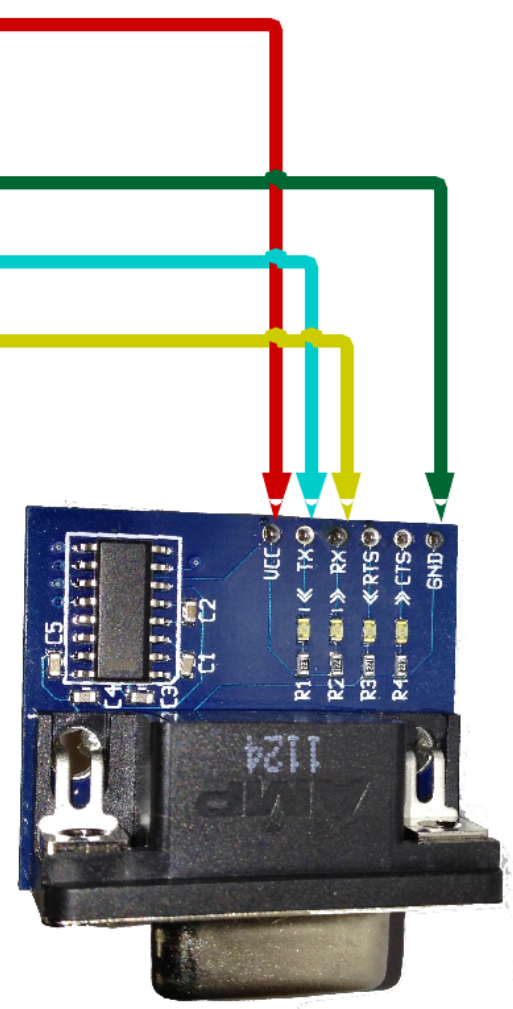
RS-232 Level Shifter

Raspberry Pi P1 Header				
PIN #	NAME		NAME	PIN #
	3.3 VDC Power	1	5.0 VDC Power	2
8	SDA0 (I2C)	3	DNC	4
9	SCL0 (I2C)	5	0V (Ground)	6
7	GPIO 7	7	TxD	15
	DNC	9	RxD	16
0	GPIO 0	11	GPIO1	1
2	GPIO2	13	DNC	14
3	GPIO3	15	GPIO4	4
	DNC	17	GPIO5	5
12	MOSI	19	DNC	20
13	MISO	21	GPIO6	6
14	SCLK	23	CE0	10
	DNC	25	CE1	11

<http://www.pi4j.com>



RS232 Controlled LED Message Sign



RS232 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.



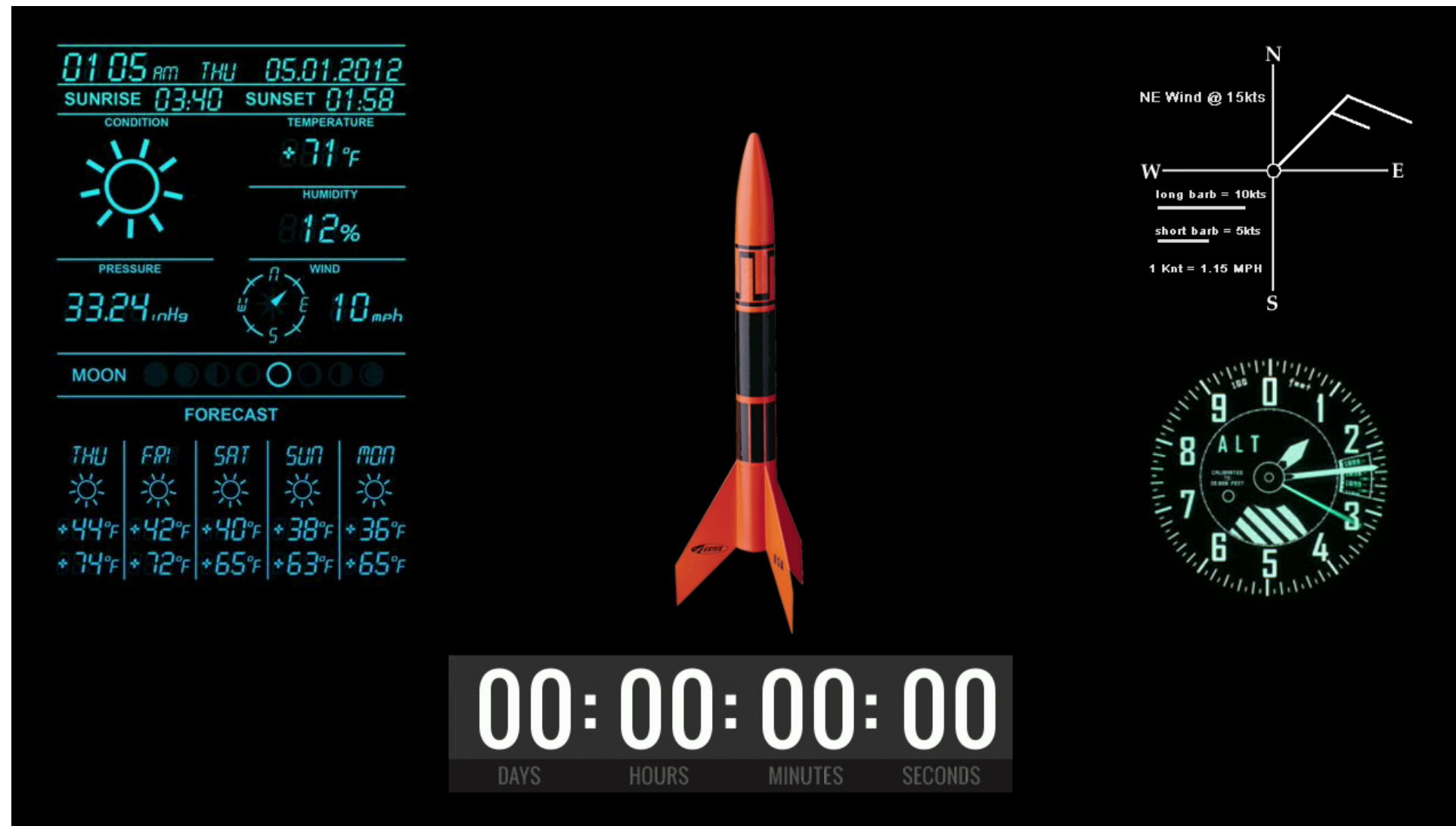
- Implement a countdown timer for the launch sequence.

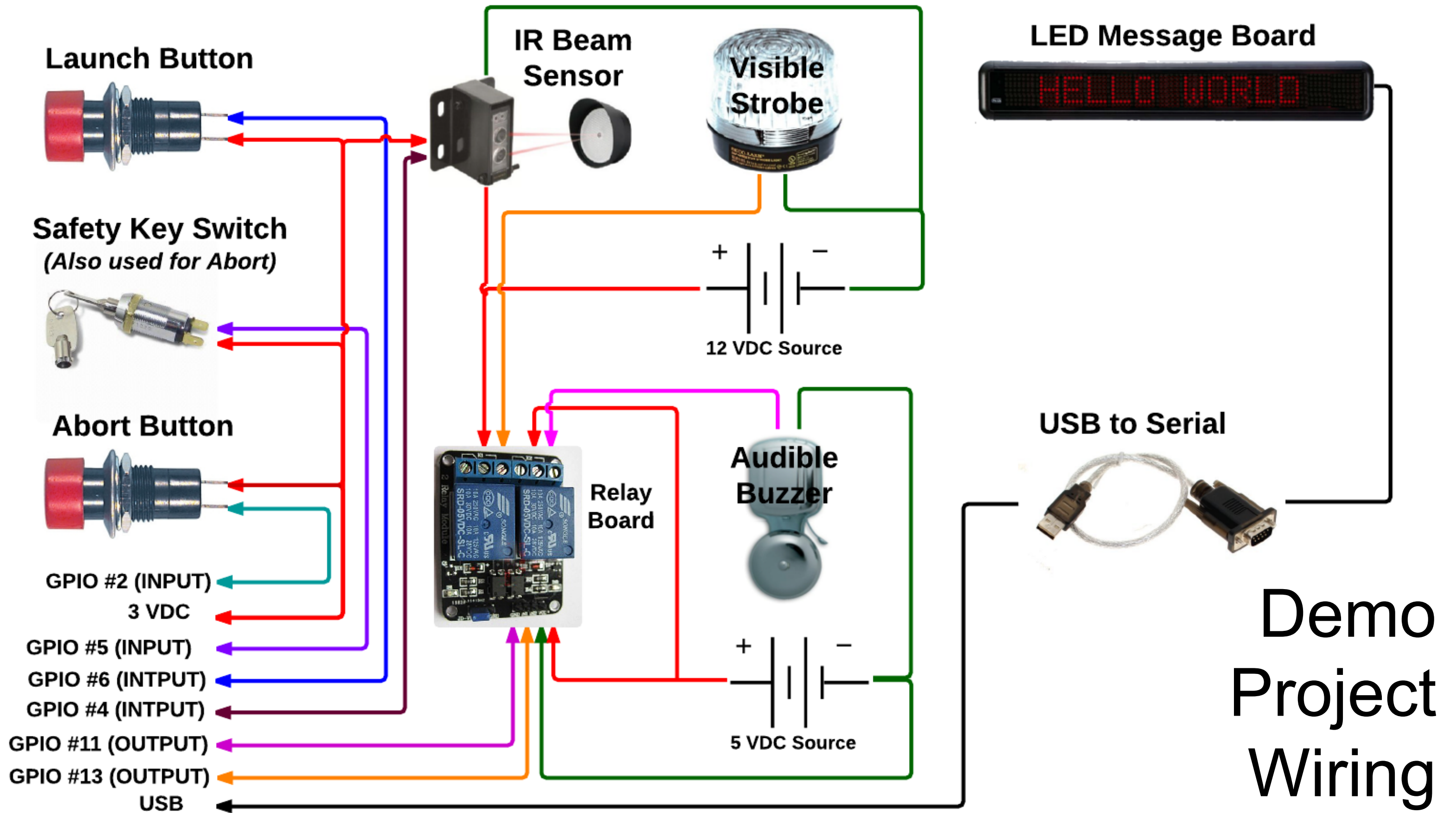




Demo Project Requirements

- Implement an on-screen console/dashboard







Demo Project

- Sample Code
- Live Demo



Savage Home Automation Blog



savagehomeautomation.com



[@savageautomate](https://twitter.com/savageautomate)



The Pi4J Project



pi4j.com



[@pi4j](https://twitter.com/pi4j)