

netcetera

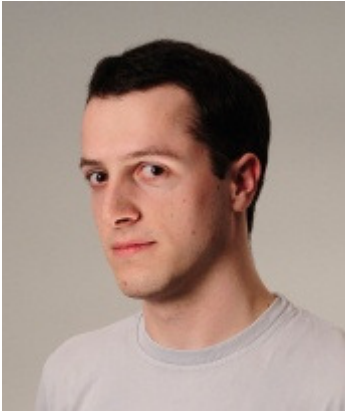
Hands on Java8 and RaspberryPi

Hacking the RaspberryPi with Java8, JavaFX8 and add-on hardware modules

25 June 2014, jug.ch – Pance Cavkovski & Aleksandar Nikov



About the speakers



Pance Cavkovski

pance.cavkovski@netcetera.com

Senior Software Engineer @ Netcetera Skopje



Aleksandar Nikov

aleksandar.nikov@netcetera.com

Head of Software Engineering @ Netcetera Skopje

About the session

What will we present

- Using Java8 and JavaFX8 on Raspberry Pi
- Extending the RasPi with components
- Interfacing with the components via Java code

You should have at the moment:

- RasPi Model B with pre-configured Raspbian on an SD Card
- A set of components (10 resistors, 10 jumper wires, 2 LEDs, 1 LDR)
- Breadboard (with a taster installed on it)
- Arduino UNO/Leonardo
- ITEAD Studio NFC module

Online source at <https://github.com/hsilomedus/hands-on-raspi>

Java8 is out!

- **Lang: Lambda, Method references, default methods, repeating annotations ...**
- **Collections: StreamAPI, HashMaps improvements**
- **Compact profiles, Security and tools improvements**
- **JavaFX8: bundled with JDK8, UI components, new theme, WebView, 3D graphics, print, css styling...**
- **Nashorn javascript engine**
- **New Date-Time, concurrency extensions, new Java DB...**
- **Linux ARM v6/v7 Hard Float ABI JDK8**

Java8 on ARM devices

Get and use with RasPi and Raspbian

- Get from <http://www.oracle.com/technetwork/java/javase/downloads>
 - Specifically Linux ARM v6/v7 Hard Float ABI
- tar xfv jdk-8....tar.gz
- ./jdk1.8.0/bin/java and ./jdk1.8.0/bin/javac
- Need hardware access? Get pi4j: <http://pi4j.com/>
 - Based on WiringPi
 - Jar(s) ready for usage

pi4j

Java library for full access to the Raspberry Pi

- JNI to the WiringPi C library
- Features
 - Export & unexport GPIO pins
 - Configure and Control GPIO pins for both input and output
 - Interrupt-based listeners
 - RS232 communication
 - I2C and SPI support
 - Extensions ...
- Both managed and direct access to WiringPi

JavaFX8

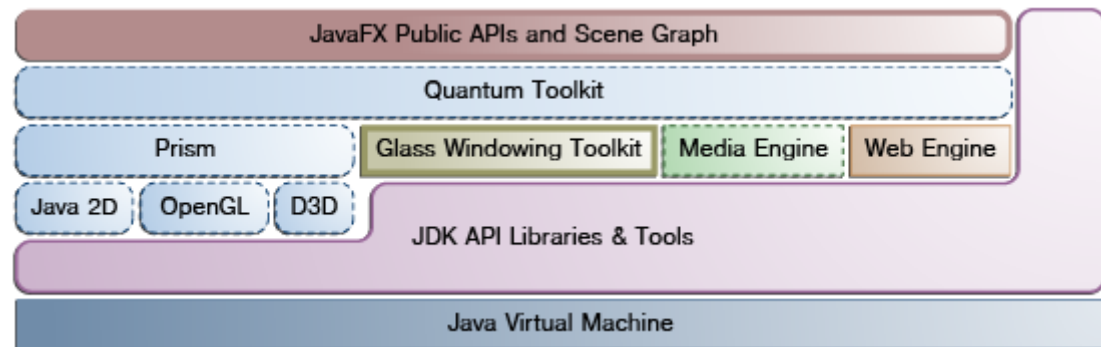
JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug and deploy rich client applications that operate consistently across diverse platforms (tl;dr: Java framework for building RIA / Desktop apps)

- Java APIs, fully integrated in JavaSE
- Declarative in FXML or explicit in Java
- Swing interoperability
- CSS stylable UI components library, Modena theme
- 3D features, hardware acceleration, web view, high-performance media engine
- Canvas & printing API, rich text support

A glimpse into the architecture

- Stage, Scene, Layouts
- Nodes, states, effects

- Hardware/software rendering



- JavaFX, Prism and Media separate threads
- Async elements update via Pulse

UI

- Layouts
- Transformations
- Effects
- CSS Styling
- Event Handlers



JavaFX8 - How To

Layout and place Components

```
primaryStage.setTitle("JavaFX Welcome");

GridPane grid = new GridPane();
grid.setAlignment(Pos.CENTER);
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(25, 25, 25, 25));

Scene scene = new Scene(grid, 300, 275);
primaryStage.setScene(scene);

Label userName = new Label("User Name:");
grid.add(userName, 0, 1);

TextField userTextField = new TextField();
grid.add(userTextField, 1, 1);
```

```
Button btn = new Button("Sign in");
HBox hbBtn = new HBox(10);
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
hbBtn.getChildren().add(btn);
grid.add(hbBtn, 1, 4);

btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent e) {
        actiontarget.setFill(Color.FIREBRICK);
        actiontarget.setText(
            "Sign in button pressed");
    }
});

primaryStage.show();
```

JavaFX8 - How To

Style with CSS

```
scene.setStyle("-fx-background-image:  
url('background.jpg');");
```

```
userName.setStyle("-fx-font-size: 12px;" +  
    "-fx-font-weight: bold;" +  
    "-fx-text-fill: #333333;" +  
    "-fx-effect: dropshadow( gaussian ,  
    rgba(255,255,255,0.5) , 0,0,0,1 );");
```

```
//OR:  
scene.getStylesheets().add  
(Login.class.getResource("Login.css").toExternalFo  
rm());
```

```
.root {  
    -fx-background-image: url("background.jpg");  
}  
.label {  
    -fx-font-size: 12px;  
    -fx-font-weight: bold;  
    -fx-text-fill: #333333;  
    -fx-effect: dropshadow( gaussian ,  
    rgba(255,255,255,0.5) , 0,0,0,1 );  
}
```

JavaFX8 - How To

Declarative

```
Parent root = FXMLLoader.load(getClass().getResource("fxml_example.fxml"));
```

```
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>
<GridPane fx:controller="fxmlexample.FXMLExampleController"
  xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">

  <padding><Insets top="25" right="25" bottom="10" left="25"/></padding>

  <Label text="User Name:"
    GridPane.columnIndex="0" GridPane.rowIndex="1"/>
  <TextField
    GridPane.columnIndex="1" GridPane.rowIndex="1"/>
</GridPane>
```

However...

The ARM devices are slower. Be careful

- RasPi has a 700MHz v7 ARM processor with 512MB ram.
- The Embedded Full Size Operating System are still catching on
- Runs in FrameBuffer and not in X

Not really plug & play:

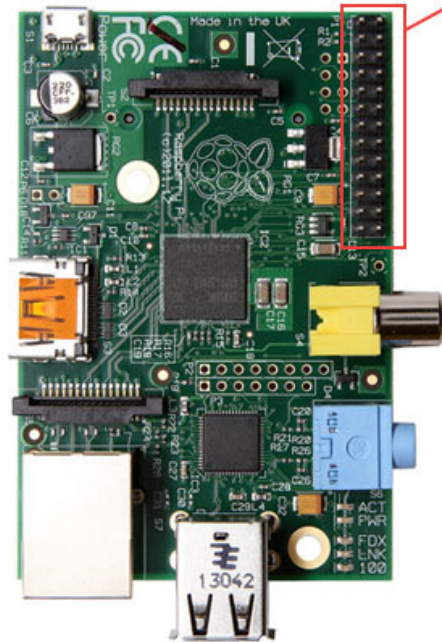
- - sudo jdk1.8.0/bin/java -cp ./:pi4j-core.jar:jdk1.8.0/jre/lib/jfxrt.jar*
-Djavafx.platform=eglfb* Main

* (not needed in the last Raspbian version)

RasPi GPIO header

Raspberry Pi Model B
shown below

GPIO pins



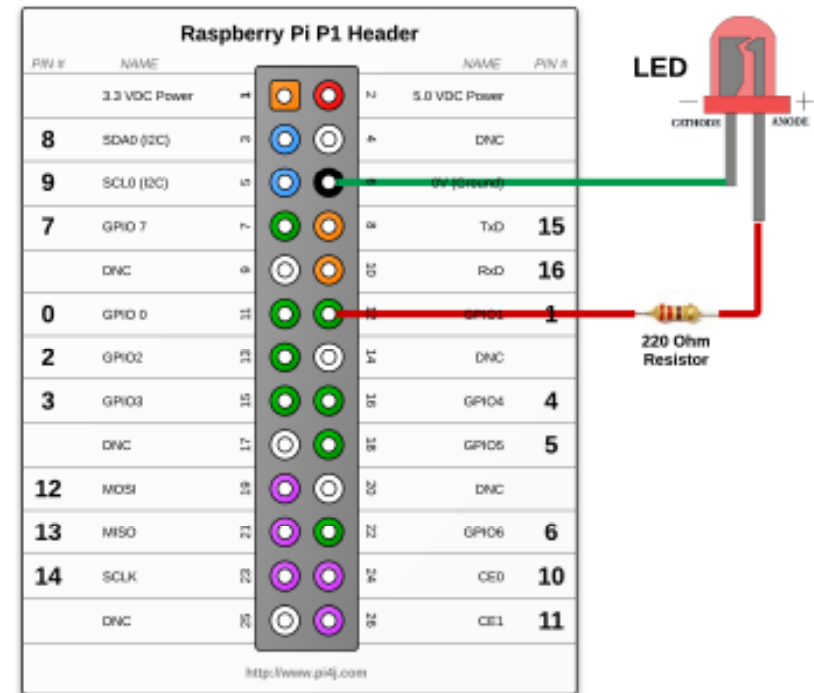
Raspberry Pi Model A				
3.3V	1	2	5V	
I2C0 SDA	3	4	DNC	
I2C0 SCL	5	6	Ground	
GPIO 4	7	8	UART TXD	
DNC	9	10	UART RXD	
GPIO 17	11	12	GPIO 18	
GPIO 21	13	14	DNC	
GPIO 22	15	16	GPIO 23	
DNC	17	18	GPIO 24	
SP10 MOSI	19	20	DNC	
SP10 MISO	21	22	GPIO 25	
SP10 SCLK	23	24	SP10 CE1 N	
DNC	25	26	SP10 CE1 N	

Raspberry Pi Model B				
3.3V	1	2	5V	
I2C1 SDA	3	4	5V	
I2C0 SCL	5	6	Ground	
GPIO 4	7	8	UART TXD	
Ground	9	10	UART RXD	
GPIO 17	11	12	GPIO 18	
GPIO 27	13	14	Ground	
GPIO 22	15	16	GPIO 23	
3.3V	17	18	GPIO 24	
SP10 MOSI	19	20	Ground	
SP10 MISO	21	22	GPIO 25	
SP10 SCLK	23	24	SP10 CE1 N	
Ground	25	26	SP10 CE1 N	

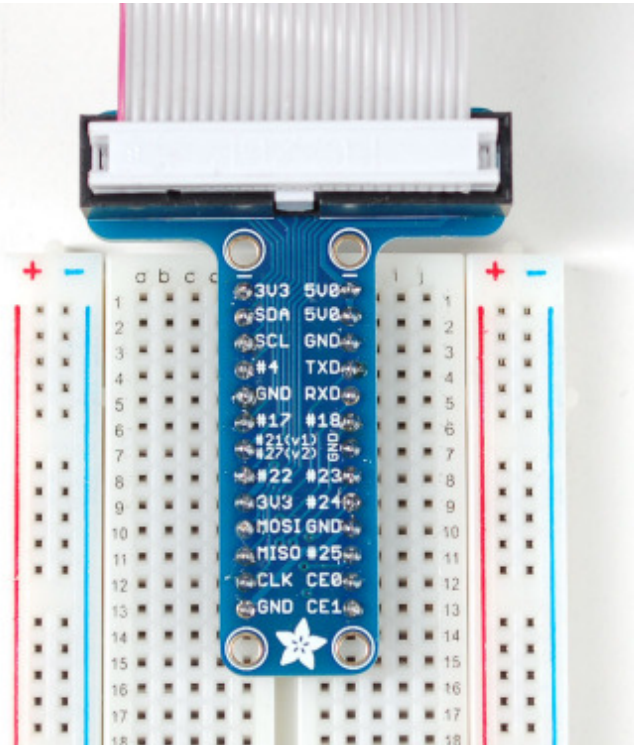
RasPi GPIO header

Physical / Raspberry Pi Name		
3.3v	1	2 5v
SDA0	3	4 DNC
SCL0	5	6 0v
GPIO 7	7	TX
DNC	9	10 RX
GPIO 0	11	12 GPIO 1
GPIO 2	13	14 DNC
GPIO 3	15	16 GPIO 4
DNC	17	1 GPIO 5
SPI MOSI	19	20 DNC
SPI MISO	21	22 GPIO 6
SPI SCLK	23	24 SP10 CEO N
DNC	26	26 SP10 CE1 N

Broadcom names		
3.3v	1	2 5v
I2C0 SDA	3	4 DNC
I2C0 SCL	5	6 0v
GPIO 4	7	UART TXD
DNC	9	10 UART RXD
GPIO 17	11	12 GPIO 1
GPIO 21	13	14 DNC
GPIO 22	15	16 GPIO 23
DNC	17	1 GPIO 24
SPI MOSI	19	20 DNC
SPI MISO	21	22 GPIO 25
SPI SCLK	23	24 SP10 CEO N
DNC	26	26 SP10 CE1 N



RasPi GPIO header cobbler and protoboard



Examples

- LED Blink
 - Taster + LED Blinker
 - Serial Comm + Arduino + Light Sensor + LED
 - NFC Reader
 - JavaFX LED and Taster
-
- Examples stored as git branches
 - Source code available at <https://github.com/hsilomedus/hands-on-raspi>

LED Blink

Checkout

Open terminal

- `cd hands-on-raspi/`
- `git checkout master`

- Open the Main source file for inspection
 - `[leafpad | vi | nano] src/Main.java`

LED Blink

```
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;

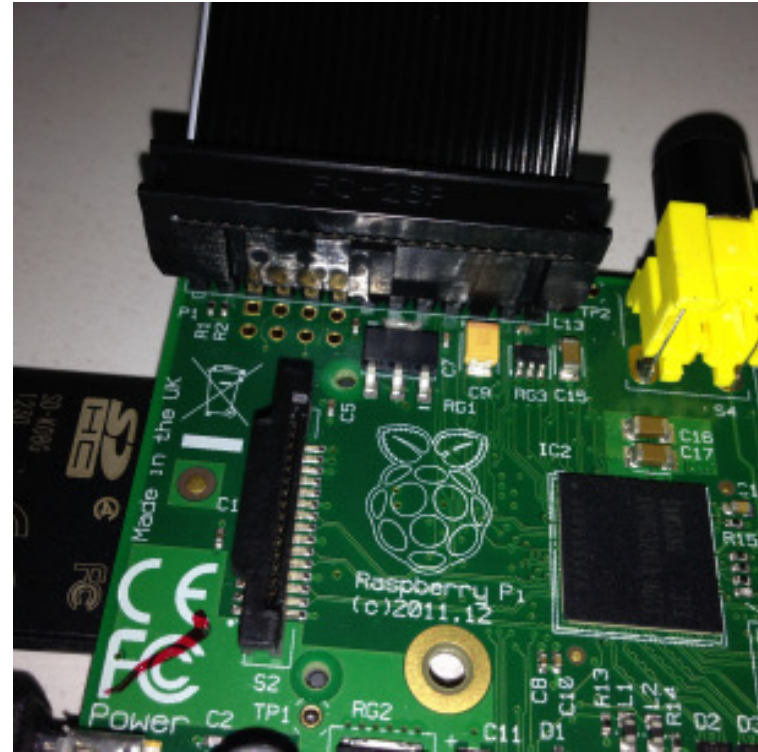
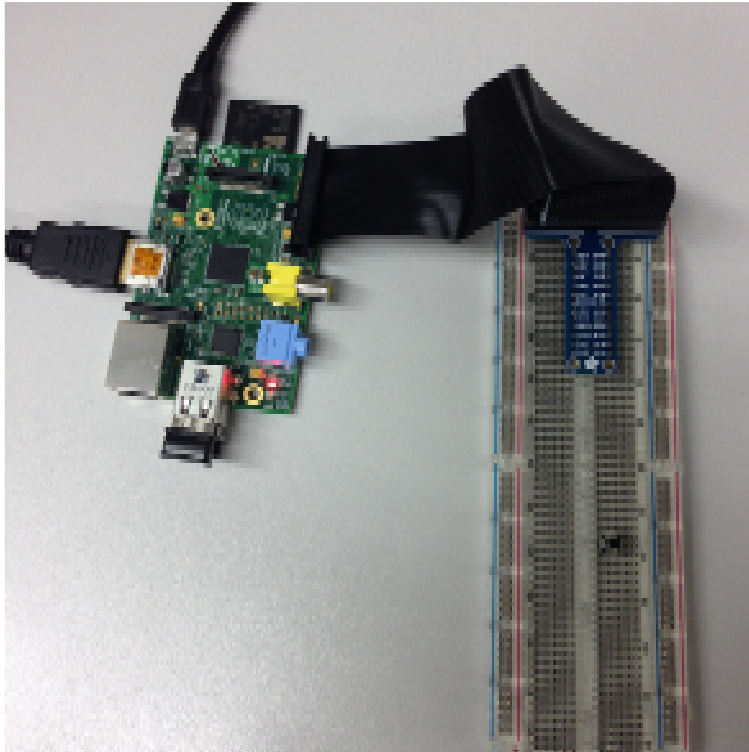
public class Main {

    public static void main(String[] args) throws InterruptedException {
        GpioController gpio = GpioFactory.getInstance();

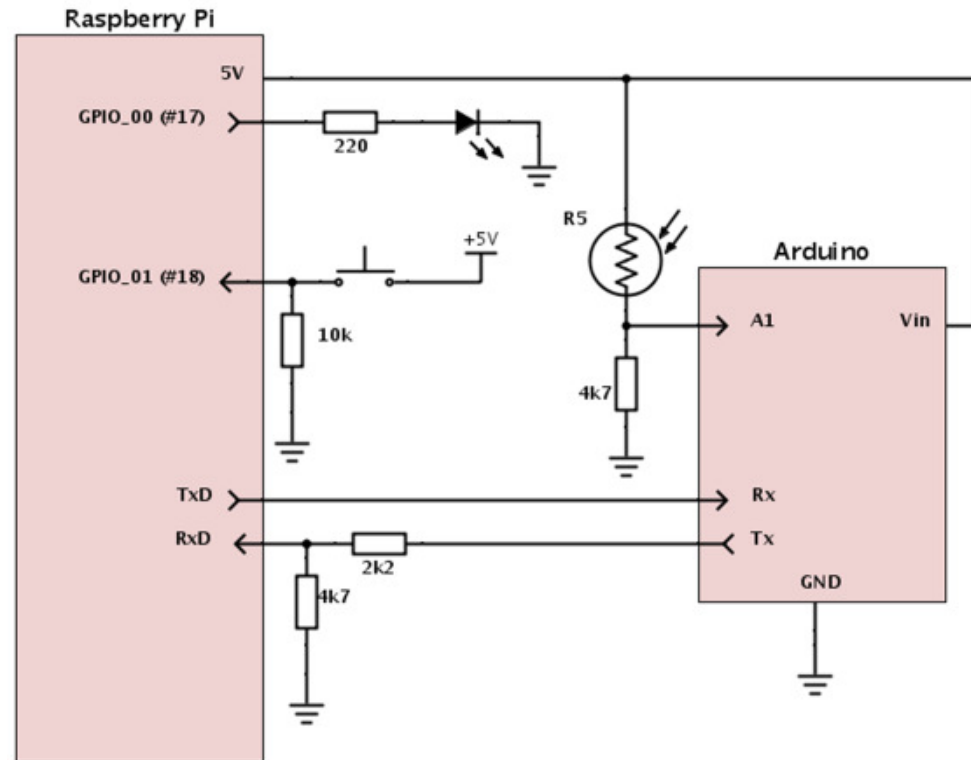
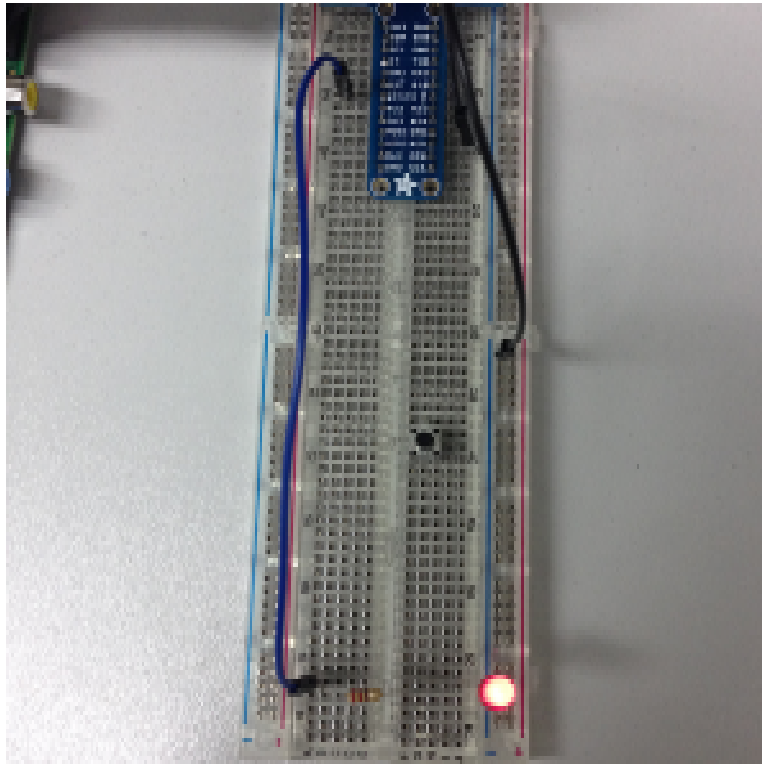
        System.out.println("Pin going UP!");
        final GpioPinDigitalOutput pin = gpio.provisionDigitalOutputPin(
            RaspiPin.GPIO_00, "LED", PinState.HIGH);

        Thread.sleep(5000);
        System.out.println("Pin going down.");
        pin.low();
        gpio.shutdown();
    }
}
```

Connect the breadboard



LED Blink



LED Blink

Build and run

- ./build.sh
 - Create/clean target dir
 - Copy libs
 - Execute javac with added pi4j into classpath
- ./run.sh
 - Sudo execute java with added pi4j into classpath

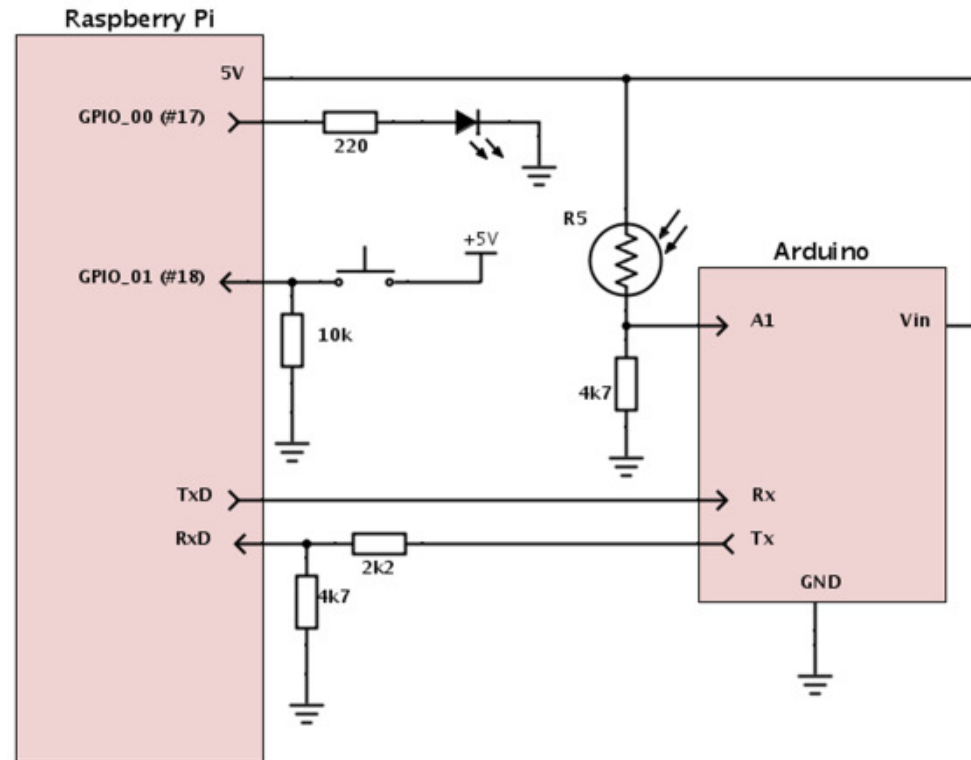
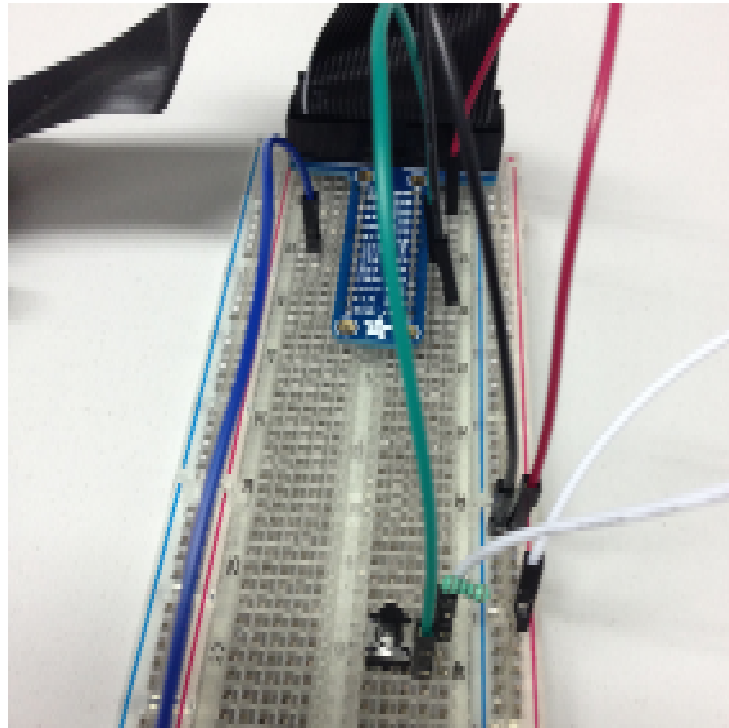
Taster + LED Blinker (git checkout taster)

```
final GpioPinDigitalInput pinInput = gpio.provisionDigitalInputPin(
    RaspiPin.GPIO_01, PinPullResistance.PULL_DOWN);
pinInput.addListener(new GpioPinListenerDigital() {
    @Override
    public void handleGpioPinDigitalStateChangeEvent(
        GpioPinDigitalStateChangeEvent event) {
        if (event.getState().equals(PinState.HIGH)) {
            System.out.println("Taster clicked!");
            change = !change;
        }
    }
});

while (true) {
    Thread.sleep(200);

    if (change) {
        pin.toggle();
    }
}
```

Taster + LED Blinker



Taster + two blinkers

Extend

- Use `RaspiPin.GPIO_02`
- Connect jumper cable + another 220 Ohms + Green LED in serial
- Code
 - Make the LEDs to blink alternatively
 - Make the green LED to light up while the taster is down.

Serial Comm + Arduino + Light Sensor + LED

- Arduino
 - Open-source electronics prototyping platform
 - Programmable IC on board with header pins
- Serial Communication
 - Lightweight two-way communication between two devices
 - Needs two connections: Rx1 to Tx2 and Rx2 to Tx1
 - Needs compatible baud-rate (9600bps), data/stop/parity bits (8-N-1) and voltage
- Light Sensor
 - Variable resistor based on light
 - <1K on intense light, ~4k7 on daylight, > 70K on dark

Serial Comm (git checkout serial) - Arduino code

```
int led = 13;

//if UNO:
HardwareSerial serial = Serial;

//if LEONARDO:
//HardwareSerial serial = Serial1;

void setup() {
  serial.begin(9600);
  pinMode(led, OUTPUT);
}
int readValue=0;

void loop() {

  readValue = analogRead(A1);
  serial.print(" ");
  serial.println(readValue,DEC);
  serial.flush();
  delay(1000);
}

void serialEvent() {
  while (serial.available()) {
    serial.read();
  }
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(50); // wait for a second
  digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
}
}
```

Serial Comm (git checkout serial) – pi4j code

```
Serial serial = SerialFactory.createInstance();

serial.addListener(new SerialDataListener() {

    @Override
    public void dataReceived(SerialDataEvent event) {
        String readString = event.getData().trim();
        System.out.println("Data received: " + readString);
        try {
            int value = Integer.parseInt(readString);
            lastValue = value;
        } catch (Exception ignore) {

        }

        //alter the state
        boolean newState = lastValue < 200;
        if (newState != outState) {
            outState = newState;
            pinOutput.setState(outState);
        }

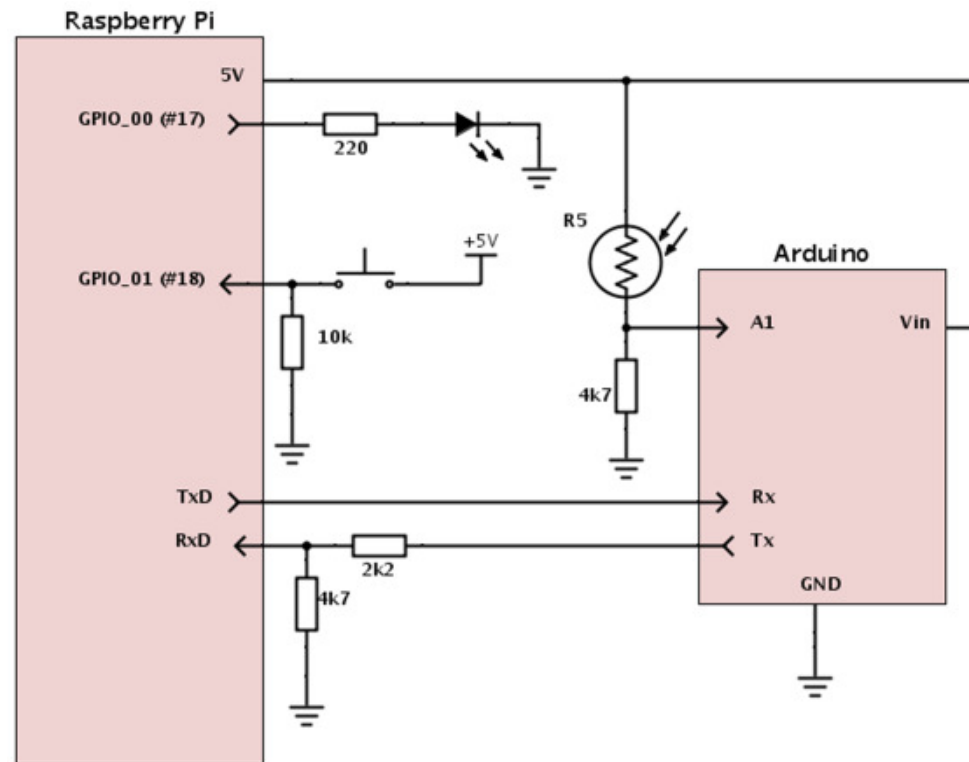
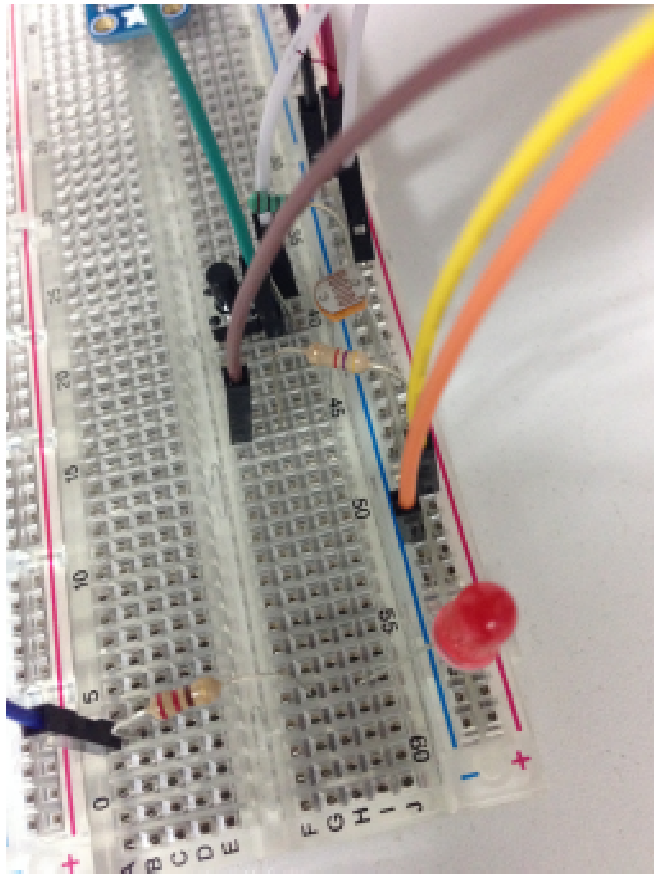
    }

});

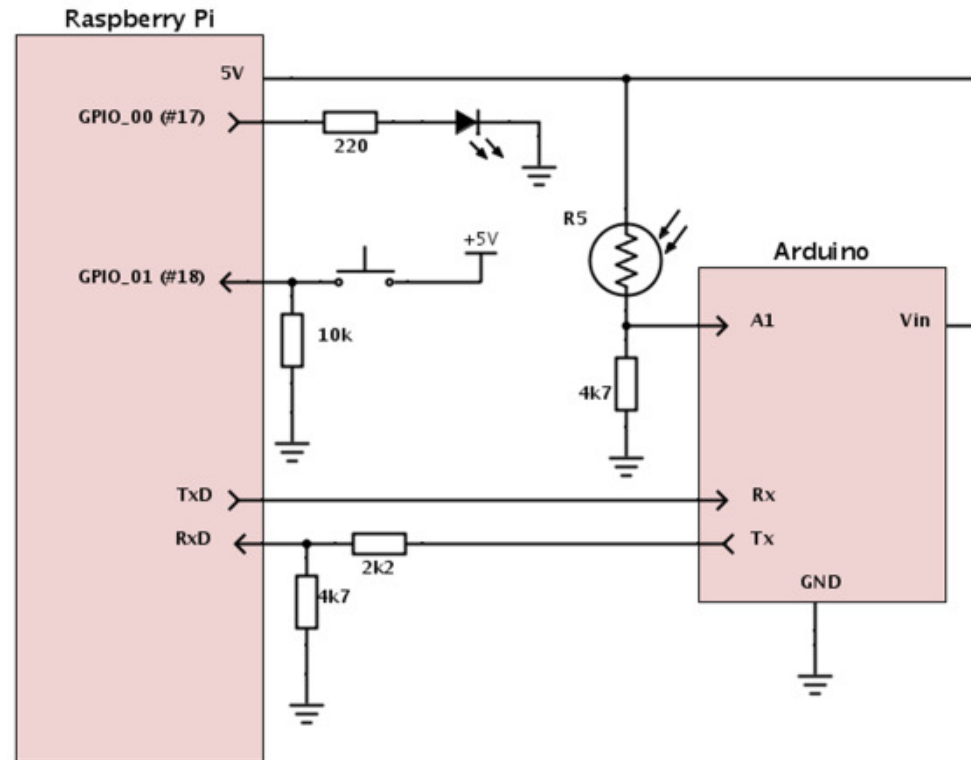
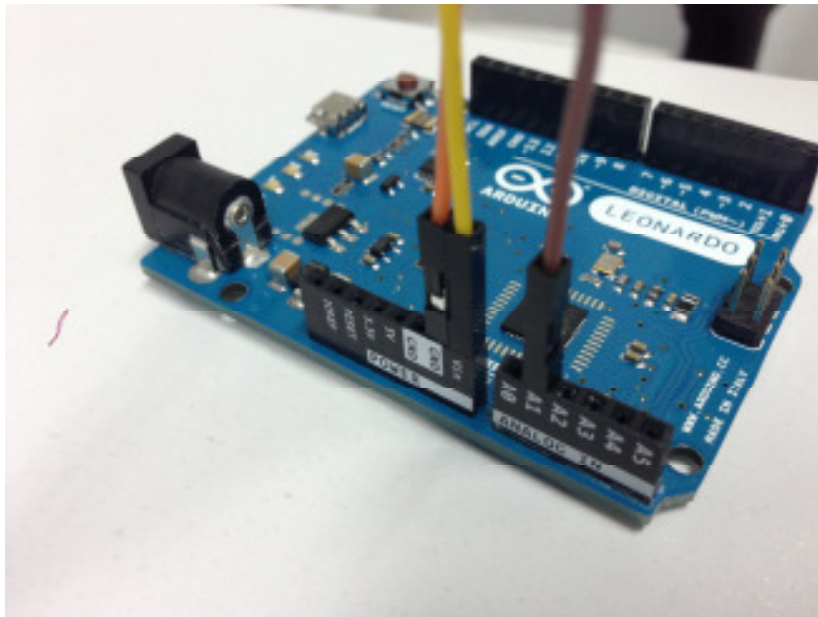
serial.open(Serial.DEFAULT_COM_PORT, 9600);

while (true) {
    System.out.println("Sent to serial.");
    serial.writeln("Text");
    serial.flush();
    Thread.sleep(2000);
}
```

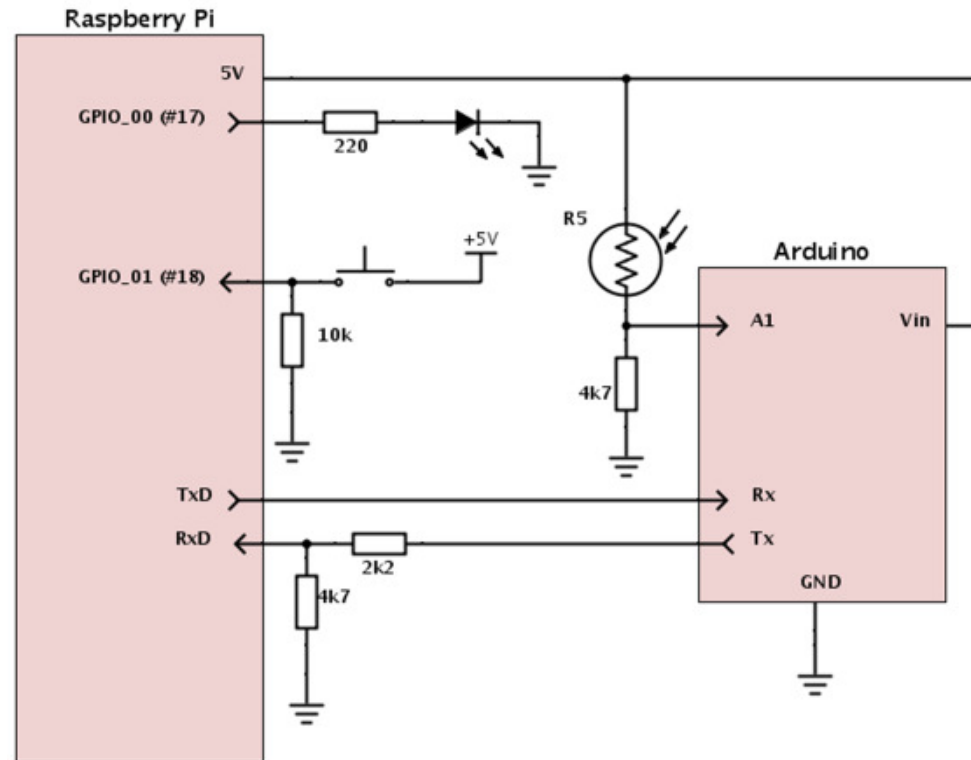
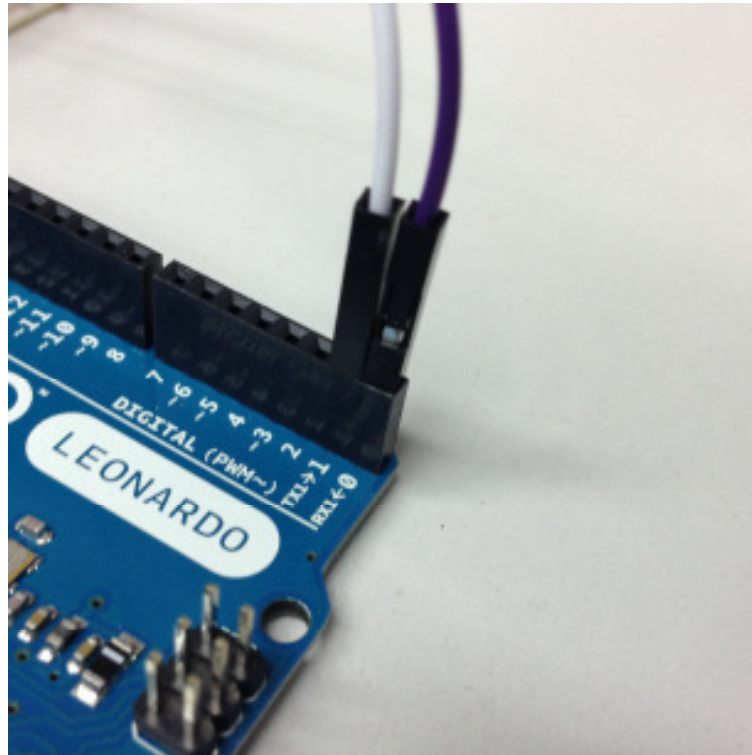
Serial Comm (board connect)



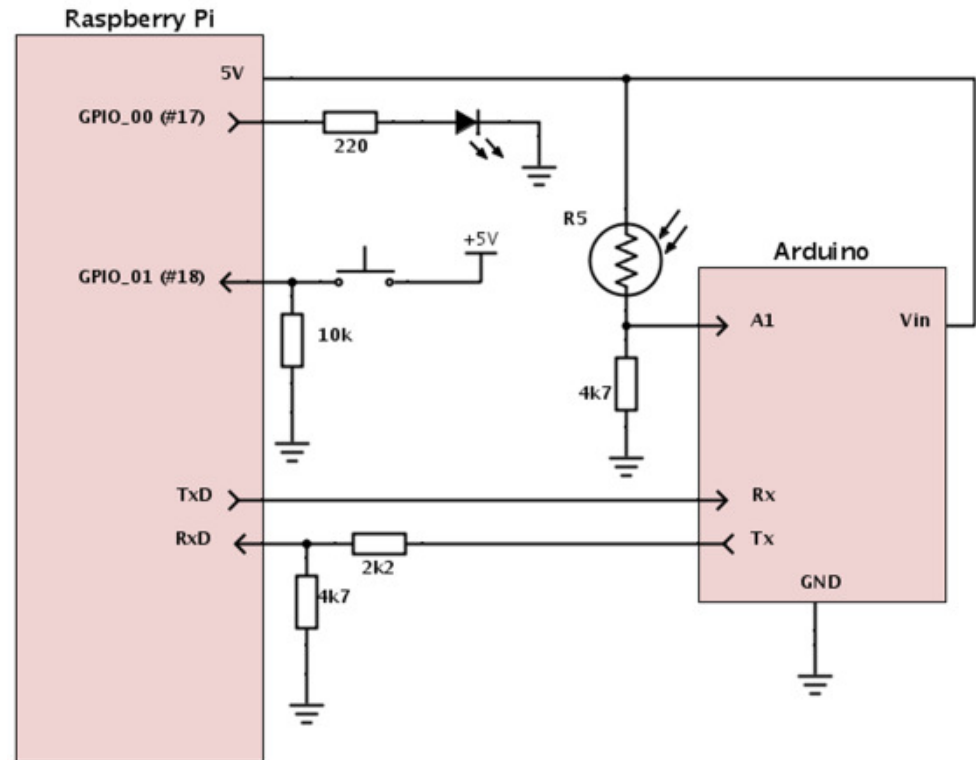
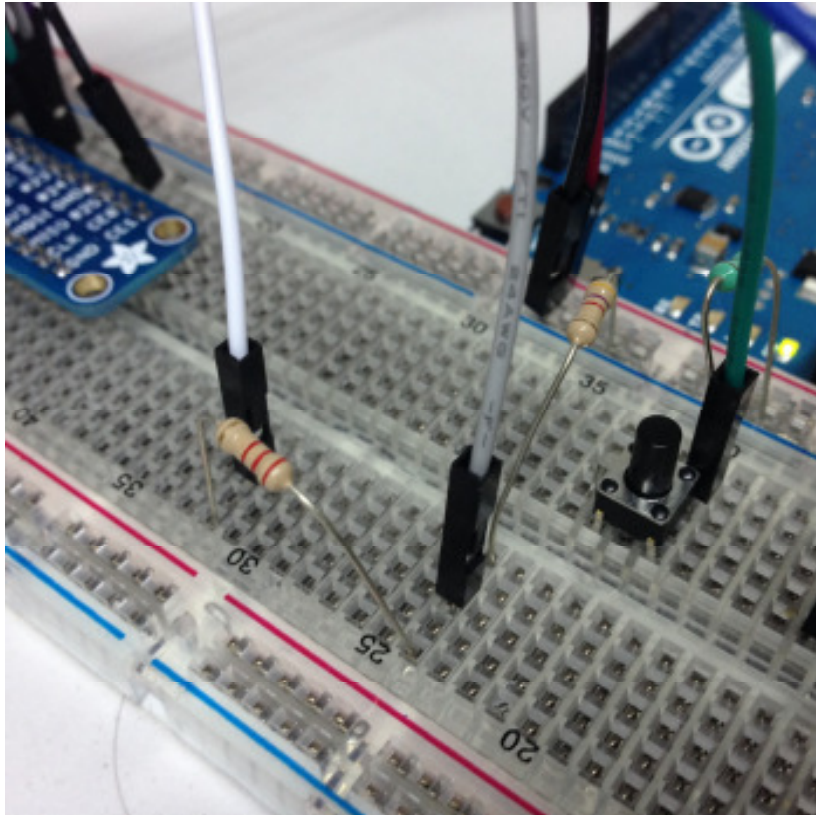
Serial Comm (board connect)



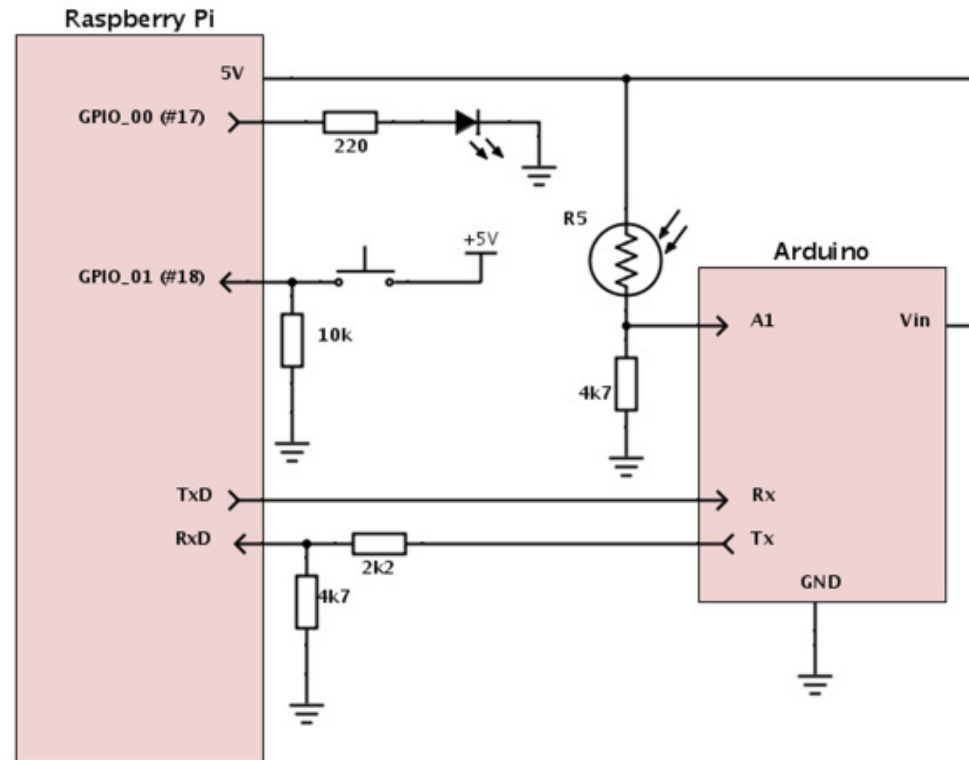
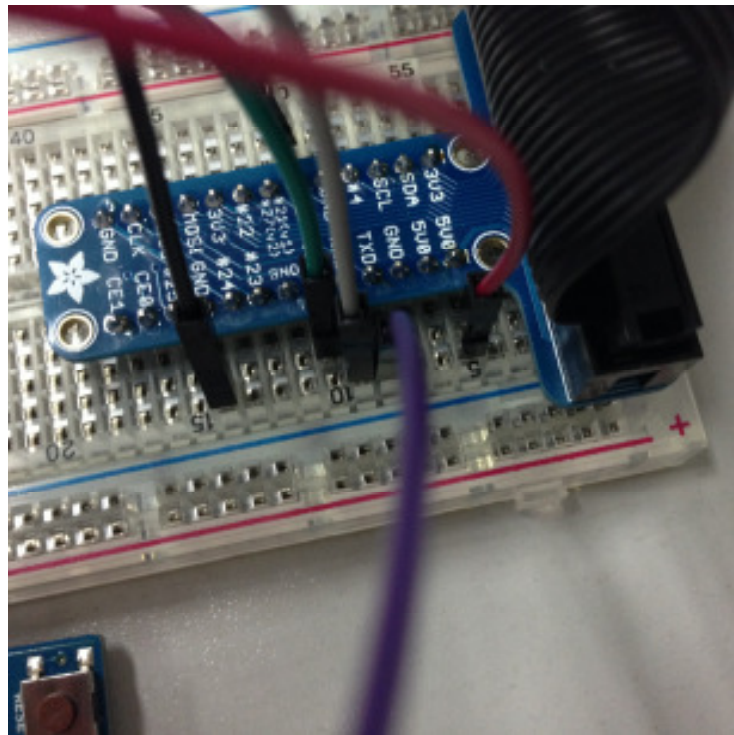
Serial Comm (board connect)



Serial Comm (board connect)



Serial Comm (board connect)



Serial Comm

Execute

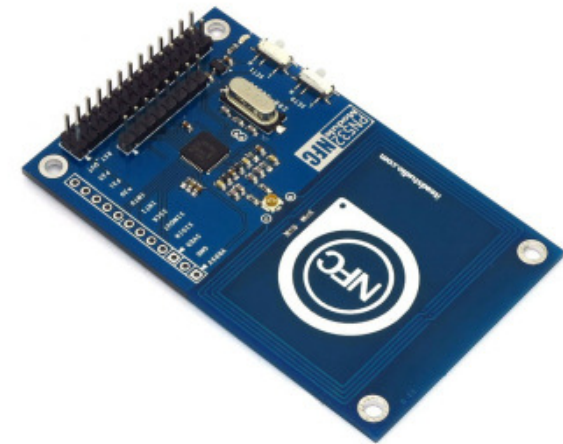
- Build & run
- Inspect the standard output
- Illuminate the sensor with your phone
- Cover the sensor with your finger

- Alternative:
 - RasPi will recognize input $< 1\text{ V}$ as a logical zero, $> 2\text{V}$ as a logical one
 - “Calculated” voltage divider can do the trick

NFC Reader

ITEAD PN532 NFC Module

- <http://imall.iteadstudio.com/im130625002.html>
- 13.56 MHz Near Field Communication
- Onboard antenna
- Up to 3 CM effective distance
- Compatible with ISO14443 Type A and B
- Supports SPI, I2C and UART interface
- With exact RasPi GPIO layout
 - Connects with the same ribbon cable
- Provided Arduino and RasPi libraries (in C)



NFC Reader

git checkout nfc-serial

- **Inspect Main.java, PN532.java and PN532Spi.java**
- Follows strict protocol on byte level
 - Begin()
 - Init wiringPi and wiringPiSpi with channel and speed
 - Wakeup()
 - Fire up CS falling edge
 - Write Command()
 - Send boxed command, data, computed checksum, wait for and check ACK
 - ReadResponse()
 - Read bytes and store in internal buffer

NFC Reader

git checkout nfc-serial

- Various commands
 - `getFirmwareVersion()` [0x02]
 - `SAMConfig()` [0x14, 0x01, 0x14, 0x01]
 - `readPassiveTargetID` [0x4A, 0x01, baudrate]
 - Auth/read/write Block (not implemented)
 - Used non-managed pi4j approach
 - `Gpio.wiringPiSetup()`
 - `Gpio.wiringPiSPISetup(SPICHANNEL, SPISPEED)`
 - `Gpio.pinMode(pin, INPUT/OUTPUT)` and `Gpio.digitalWrite(pin, HIGH/LOW)`
 - `Spi.wiringPiSPIDataRW(SPICHANNEL, dataToSend, 1);`

NFC Reader

- Connect
- Build & Run
- Observe



TasterFX (git checkout taster-fx)

```
Button btn = new Button();
btn.setText("Toggle LED");
btn.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent event) {
        pin.toggle();
    }
});

lab = new Label();
lab.setText("Clicks: ");

Button btnExit = new Button();
btnExit.setText("Exit");
btnExit.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent event) {
        Platform.exit();
    }
});
```

```
VBox vbox = new VBox();
vbox.getChildren().addAll(btn, lab, btnExit);

StackPane root = new StackPane();
root.getChildren().add(vbox);

Scene scene = new Scene(root, 300, 250);

primaryStage.setTitle("Hello World!");
primaryStage.setScene(scene);
primaryStage.show();
```

TasterFX

```
pinInput.addListener(new GpioPinListenerDigital() {
    @Override
    public void handleGpioPinDigitalStateChangeEvent(
        GpioPinDigitalStateChangeEvent event) {
        if (event.getState().equals(PinState.HIGH)) {
            System.out.println("Taster clicked!");
            Platform.runLater(new Runnable() {
                @Override
                public void run() {
                    lab.setText(lab.getText() + "**");
                }
            });
        }
    }
});
```


TasterFX

Execute

- Build & run
- Click on the button
- Click the taster
- Exit and see what happens

Q & A

- <https://twitter.com/hsilomedus>
- <http://hsilomedus.me/>
- pance.cavkovski@netcetera.com