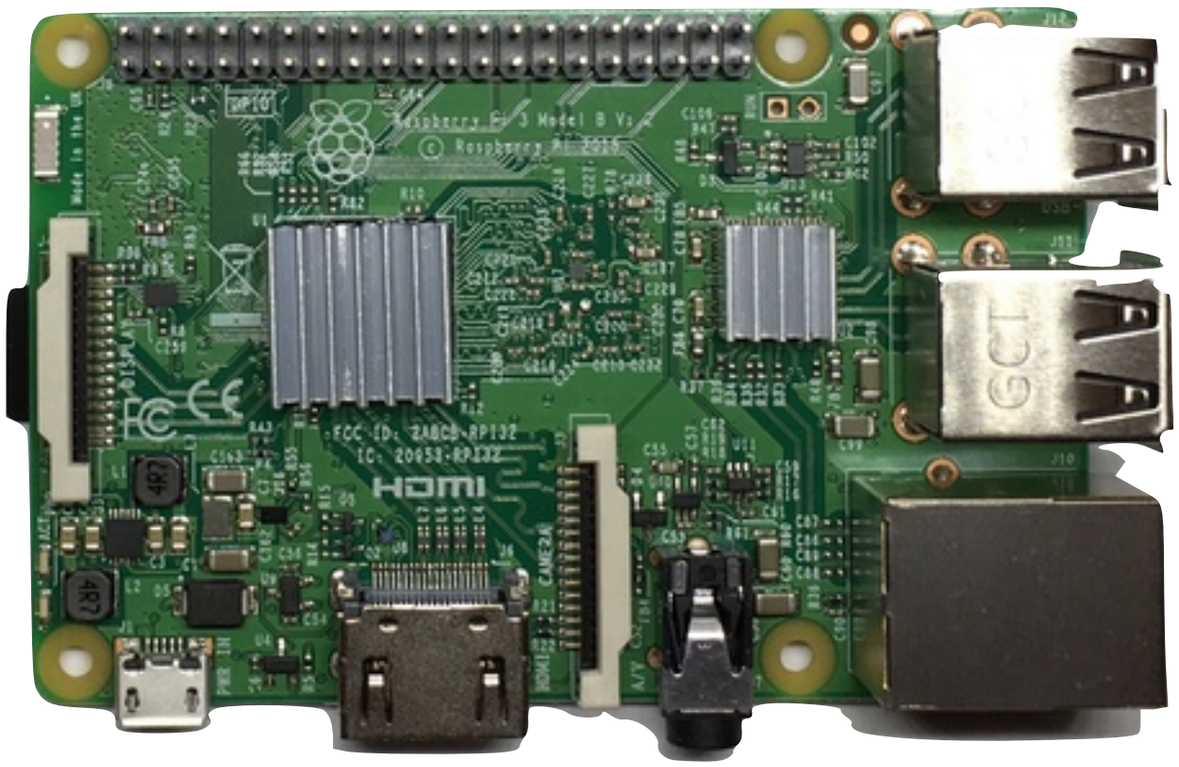
Swift and the Internet of Things

Steven Holland
stevenholland@mac.com

Arduino

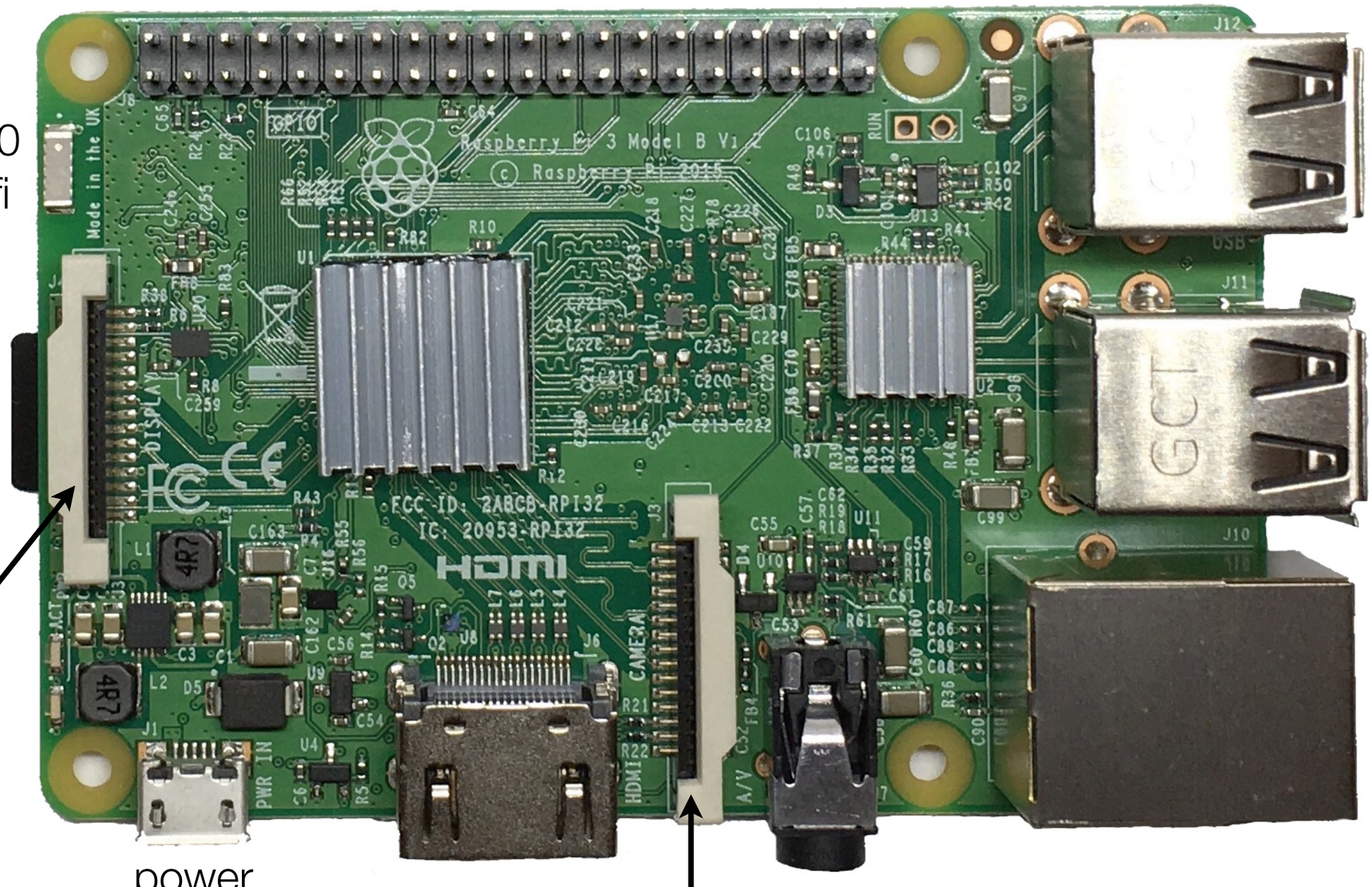


Raspberry Pi

1.2-GHz 64-bit quad-core ARMv8-A

40 GPIO pins

bluetooth 4.0
802.11n wifi



4 USB
2.0

micro
SD

DSI
display

10/100
ethernet

power
(micro USB)
5V / 2.5A

HDMI

RCA

CSI camera

Workflow options

Installing Swift

An Update on Swift 3.1.1 For Raspberry Pi Zero/1/2/3

May 1, 2017

The current status of Swift 3.1.1 on ARM-based boards like the Raspberry Pis.

[Read...]

Building a LISP from scratch with Swift

February 5, 2017

This article describes how you can build a simple LISP, based on the 1978 article *'A Micro Manual For LISP - Not The Whole Truth'* with Swift, taking advantage where possible of the features the language offers.

[Read...]

Swift 3.0.2 For Raspberry Pi Zero/1/2/3

December 30, 2016

Swift for all the Raspberry Pi boards, built on Ubuntu16.04 for RaspberryPi 2/3 and on Raspbian for the original RaspberryPis (A,B,A+,B+).

[Read...]

Installing Swift

```
sudo apt-get install clang

wget https://www.dropbox.com/s/cah35gf5ap22d11/swift-3.0.2-
RPi23-1604.tgz

tar xzf swift-3.0.2-RPi23-1604.tgz
```

also see swift.org

Testing Swift

create a Hello, World program

```
nano main.swift
```

run as a script in the REPL

```
swift main.swift
```

compile and run

```
swiftc main.swift
./main
```

Testing the GPIO pins

build the LED circuit

220 Ω resistor from G17 to LED

LED to ground

test LED from command line

```
gpio -g mode 17 out
gpio -g write 17 1
gpio -g write 17 0
```

Using the Swift Package Manager

create the app

```
mkdir blinker && cd blinker
swift package init --type executable
```

edit Package.swift to include the SwiftyGPIO package

set up for use in Xcode (if developing on Mac)

```
swift package update
swift package generate-xcodeproj
```

edit main.swift

build and run

```
swift build
.build/debug/blinker
```

also see swift.org and packagecatalog.com

Blinking an LED

```
#if os(Linux)
    import Glibc
#endif
import SwiftyGPIO
import Foundation

let pins = SwiftyGPIO.GPIOs(for: .RaspberryPi3)

guard let ledPin = pins[GPIOName.P17] else {
    fatalError("GPIO pin 17 could not be initialized")
}

ledPin.direction = .OUT
ledPin.value = 0

while true {
    ledPin.value = ledPin.value == 0 ? 1 : 0
    usleep(500000) // microseconds
}
```

Sensors

temperature
air pressure
humidity

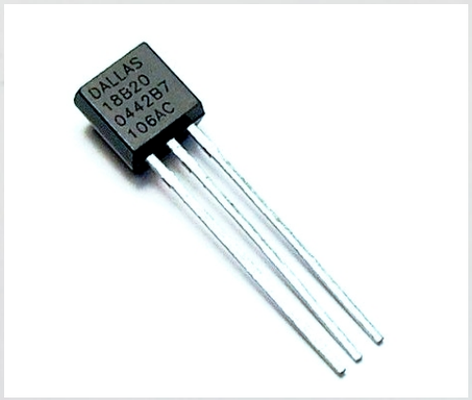
accelerometer
gps
flex
tilt
pressure
vibration
distance
Hall effect

photogate
LIDAR

pH
conductivity
liquid level
radiation
pulse
gases
dust
soil moisture
spectrometry

Temperature sensor DS18B20

green to G4
red to 5V
black to GND
470 kΩ resistor



Reading temperature sensor from command line

```
sudo nano /boot/config.txt
```

append to config.txt:

```
dtoverlay=w1-gpio
```

```
sudo reboot
```

```
cd /sys/bus/w1/devices
ls
cat 28-00000829c4da/w1_slave
```

Temperature sensor output

Temperature data

Other data

CRC expectation

CRC result

```
6a 01 4b 46 7f ff 06 10 5f : crc=5f YES
6a 01 4b 46 7f ff 06 10 5f t=22625
```

Temperature (millidegrees C)

https://cdn-shop.adafruit.com/datasheets/DS18B20.pdf

Building a Swift sensor app

create the app

```
mkdir thermometer && cd thermometer
swift package init --type executable
```

edit main.swift

build and run

```
swift build
.build/debug/thermometer
```

Extracting the temperature

```
func temperature(fromSensorString string: String) -> Double? {
    guard string.contains("YES") else {
        return nil
    }

    let trimmed = string.trimmingCharacters(in: .whitespacesAndNewlines)
    let components = trimmed.components(separatedBy: "t=")

    guard components.count == 2,
        let temperature = components.last,
        let millidegreesC = Double(temperature) else {
            return nil
    }

    return millidegreesC / 1000.0
}
```

Reading files

```
func readStringFromFile(_ path: String) -> String? {
    guard let fp = fopen(path, "r") else {
        print("WARNING: File could not be opened")
        return nil
    }
    defer { fclose(fp) }
    var outputString = ""
    let chunkSize = 1024
    let buffer: UnsafeMutablePointer<UInt8> =
      UnsafeMutablePointer.allocate(capacity: chunkSize);
      defer {buffer.deallocate(capacity: chunkSize)}
    repeat {
        let count: Int = fread(buffer, 1, chunkSize, fp)
        guard ferror(fp) == 0 else {break}
        if count > 0 {
            outputString += stringFromBytes(bytes: buffer, count: count)
        }
    } while feof(fp) == 0

    return outputString
}

func stringFromBytes(bytes: UnsafeMutablePointer<UInt8>, count: Int) -> String {
    return String((0..<count).map ({Character(UnicodeScalar(bytes[$0]))}))
}
```

Main loop

```
enum FilePath {
    static let device = "/sys/devices/w1_bus_master1/28-00000829c4da/w1_slave"
}

while true {
    if let sensorReading = readStringFromFile(FilePath.device),
        let degreesC = temperature(fromSensorString: sensorReading) {
        print("Temperature: \(degreesC) °C")
    }
    sleep(2)
}
```

Steven Holland

stratigrafia.org/blog

stevenholland@mac.com

@forktail

www.HuntMountainSoftware.com