DIY IoT (Internet of Things)

Penguicon 2017

Connie Sieh & Dave Putz

DIY IoT

Goals of this presentation:

- Give you a good understanding of IoT devices
- Help you on your way to building your own IoT device(s)
- Give you some resources to help you get more info

DIY IoT

Agenda

- Definition of some terms
- Overview of an IoT system
- Details on hardware components
- Details on software components
- Security Issues
- Example programming of an IoT device
- Available resources

DIY IoT

Why DIY?

- “Smart” lights, switches, thermostats, etc. are all commercially available
- In the future, even more devices will gain network capabilities
- Control
- Security
- “for the fun of it”

DIY IoT

- Definitions
- IoT (Internet of Things)
- Physical objects with network connectivity
- “smart” devices (lights, cars, alarms, etc.)
- Sensors, actuators
- Services and servers communicating with devices
- Estimates of 50 billion devices by 2020

–

DIY IoT

•Definitions

–MCU

•Microcontroller Unit

•Small, generally single-purpose chip

•Most often does not run an Operating System

–MPU

•Microprocessor Unit

•Usually able to multi-process

•Typically will run an Operating System

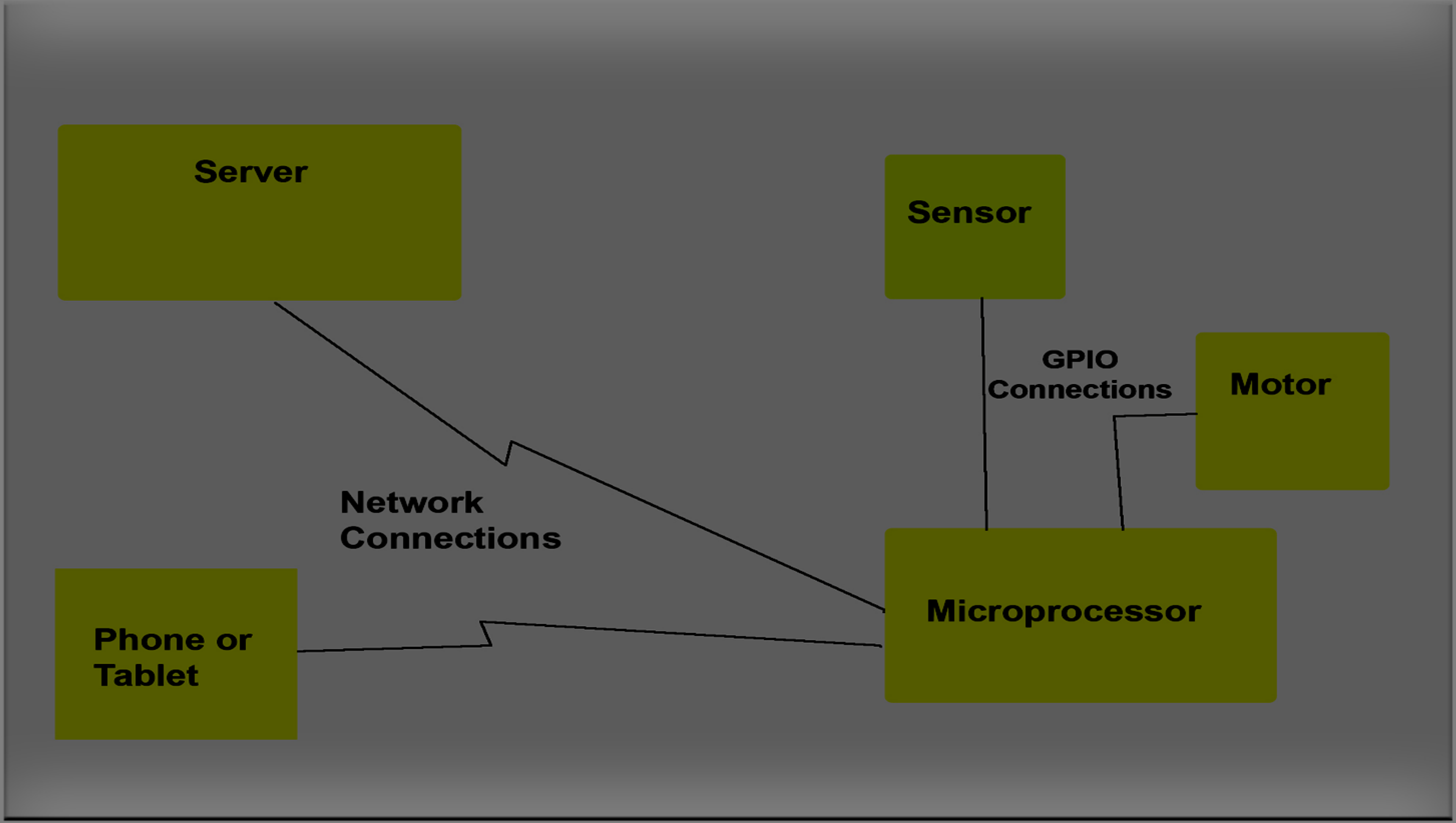
–

DIY IoT

- Overview of an IoT System
  - End devices
    - Sensors (temperature, movement, light, switches, etc.)
    - Actuators (motors, relays, LEDs, etc.)
    - Displays (optional)
  - Programmable Microprocessor
    - Able to “talk” to devices
    - Has network connectivity
    - Possibly controlled by phone, tablet, etc.
  - Network and servers
    - Data collection and analysis

DIY IoT

- Graphical Overview of a Typical IoT System



DIY IoT

- Hardware Components

–Processor

- More of an SoC “System on a Chip” than just a CPU
- Programable
- Quite a few choices available (more on this later)

–Sensors and Actuators

- Sensors report back information about the physical environment
- Actuators (motors, switches, etc.) do something in the physical environment

DIY IoT

- Hardware Components
  - Network
- Can be wireless or wired
- Differentiates IoT from previous sensors/actuators
- Various sorts of wireless available
  - Wifi
  - Bluetooth
  - LoRa Radio
  - Wemo
  - Zigbee
  - Zwave

DIY IoT

- Processor Types (MCU and MPU)
  - MPUs often have more direct compiler support on-board, MCUs typically require an external programmer.
  - Many different vendors and varieties available
- Wikipedia lists 32 MCU and 65 MPU makers
- Many vendors offer more than one model
  - We will cover the features of just a few models
  - Often the biggest single decision when designing an IoT system

DIY IoT

•Some Common Processor Types

–MCUs

•Arduino

–Pluses

•Cheap (unless networking is added)

•Lots of GPIO pins

•Relatively Low Power Required (possible with just batteries)

–Minuses

•No network without adding an additional board (called a shield)

•Fairly slow CPU speed

DIY IoT

- Some Common Processor Types

–MCUs

- ESP8266 (Many form factors)

–Pluses

- Built-in Wifi networking
- Cheap

–Minuses

- Limited number of GPIO pins
- Programming may have to allow for network activity (just 1 CPU))
- ESP32

DIY IoT

- Processor Types
  - MPUs
    - Raspberry Pi
      - Pluses
        - On-board compilers
        - Can run other software at the same time
        - Very large online community
      - Minuses
        - Limited numbers of GPIO and other pins
        - May be more expensive than an MCU
        - Configuring I2C and SPI may be more complex

DIY IoT

•Processor Types

–MPUs

•Beagleboard

–Pluses

•More powerful processor

•More GPIO pins, I2C bus pins, PWM

•Linux

–Minuses

•More expensive

•MT7688 by Mediatek (Omega2, Omega2+)

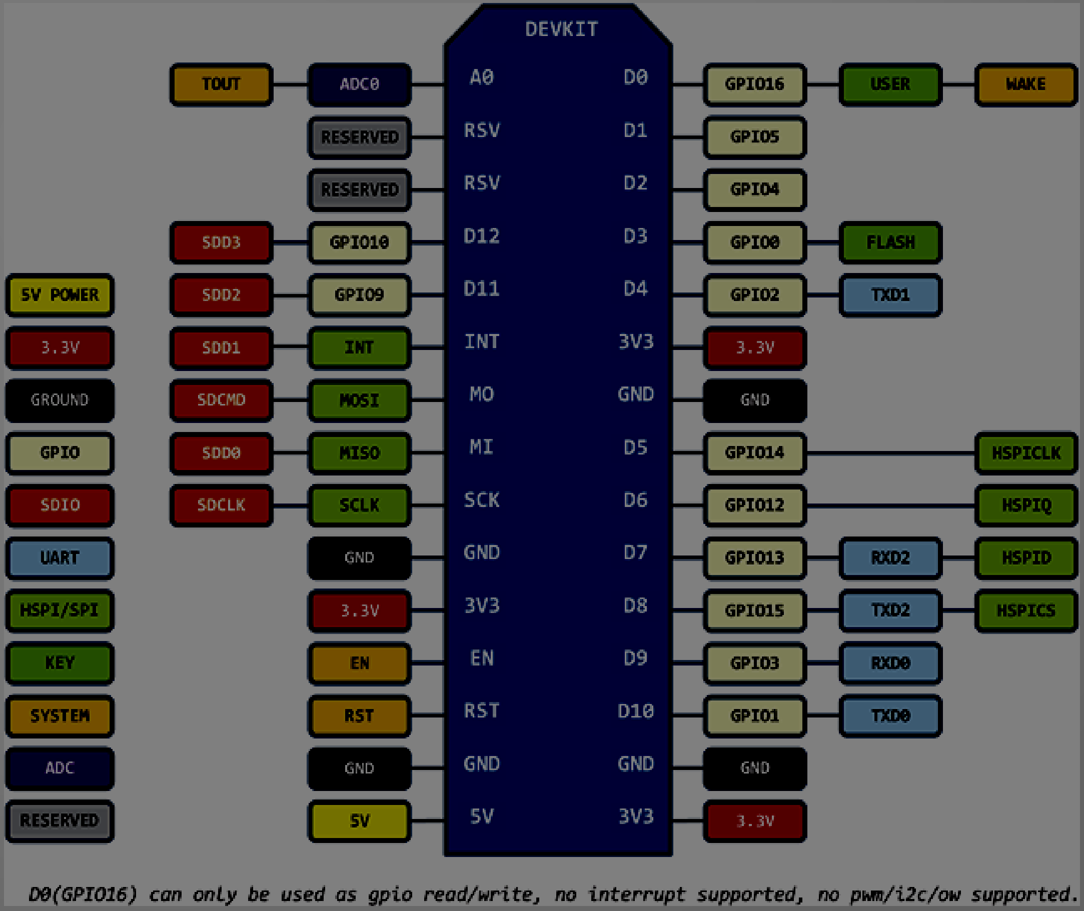
–Pluses

DIY IoT

- Sensors and Actuators
  - Use GPIO (General Purpose Input Output) pins from the CPU
  - Various protocols may be supported
- I2C – Can handle multiple devices on one set of 2pins
- UART – Universal Asynch Receive/Transmit
  - Serial port, needs two pins (one XMIT, one RCV)
  - Not all GPIO pins will support this
- SPI – Serial Peripheral Interface
  - Needs 4 pins, likely defined by hardware
- PWM – Pulse Width Modulation

DIY IoT

- GPIO Ports matter
  - You will need to check the specs for your exact chip
  - NodeMCU diagram



D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.

DIY IoT

- Some Commercially Available Devices
  - Sonoff (https://www.itead.cc/sonoff-wifi-wireless-switch.html)
- Smart switches
- ESP8266 based
- Uses Android/IOS app
- Hackable by solding pins so can upload new code
  - Alexa (http://alexa.amazon.com)
- Voice activated
- Able to interface with many of the smart devices on the market
  - Nest thermostats (https://nest.com)

DIY IoT

- Issues when selecting external devices
  - Power requirements
    - May be 3.3 or 5 volts, may not match your CPU
    - May require more current than your board can provide (i.e. could need an external power supply)
  - Connectivity
    - May require special cables
    - May need more GPIO pins than you have available
  - Openness
    - May have proprietary issues, making it hard to modify to do what you want

DIY IoT

- Software Components
  - How devices get programmed
- More capable processors (such as Raspberry Pi) may be directly programmable (i.e. have onboard compilers)
- Smaller processors (Arduino, ESP8266, etc.) require an external system to compile/download programs
  - Several languages available
- Arduino IDE uses C/C++
- Micro Python available on more capable processors
- Lua script for ESP boards
- Arduino has some visual block editors available (ArduBlock, Snap4Arduino)

DIY IoT

- Software Components
  - Software is needed external to your IoT device
- Somewhere to send monitor readings
  - MQTT broker
  - IFTTT (if this then that)
  - Flow controller (such as Node-Red)
  - Browser client
- Someone to send info that triggers an action to take
  - MQTT broker
  - Flow controller
  - Browser

DIY IoT

- Definitions
  - Message Broker (MQTT)
- Program that handles the queuing up of input messages (publishing) and sending out to interested clients (subscribing)
- Several are available, we will be using an Open Source one named mosquitto
- Fairly low overhead, can be run on small systems (such as a Raspberry Pi)
  -

DIY IoT

- Software Components
  - Protocols
- TCP/IP underlies almost everything you will do
  - Have to create or obtain an IP address
  - TCP/IP routing matters
- Advanced Message Queuing Protocol (AMQP)
- Constrained Application Protocol (CoAP)
- Extensible Messaging and Presence Protocol (XMPP)
- Message Queuing Telemetry Transport (MQTT)
- HTTP

DIY IoT

- Software Components
  - Frameworks
- May help you get running without doing much (or any) coding
- Several available
  - ESPEasy - https://www.letscontrolit.com/
  - ESPurna - https://bitbucket.org/xoseperez/espurna
  - Espidf - http://docs.platformio.org/en/latest/frameworks/espidf.html

DIY IoT

- Software Components
  - Some Additional Elements
- NodeRED
  - drag and drop visual flow control, runs on a server (Linux,Windows, Mac)
- IFTTT (If this then that) - https://ifttt.com/
  - Create applets to tie services together
- MQTT Brokers
  - Mosquitto – easy to set up, runs on a server (Linux, Windows, Mac)
  - io.adafruit.com – run by Adafruit, provides a dashboard

DIY IoT

- Security Issues
  - Unencrypted wireless network traffic can be seen by anyone close by
- Do you really want a stranger to be able to open your garage door?
  - Default router passwords are known to all the 'bad guys'
  - Vendors of commercial IoT devices have been known to be lax when it comes to security
- Google for “IoT lightbulb security” - scary
  - Bottom line – don't trust anyone else to make your devices secure

DIY IoT

- Security Issues
  - OTA (Over the air updates)
  - Using a DIY IoT system for home security
- Need to consider the monitoring issue
  - Commercial vendors are 24x7; you will need to sleep
- You control who sees what
- Several good open source packages available
  - ZoneMinder - https://zoneminder.com/
  - openhab - https://www.openhab.org/
  - Home Assistant - https://home-assistant.io/

DIY IoT

- Example Programming
  - Simple ESP8266 program

```
/*
 ESP8266 Blink by Simon Peter
*/
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on by making the voltage LOW
  delay(1000);                      // Wait for a second
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);                      // Wait for two seconds (to demonstrate the active low LED)
}
```

DIY IoT

- Example Programming
  - Simple ESP8266 program with networking

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

const char* ssid = "........";
const char* password = "........";

ESP8266WebServer server(80);
const int led = 13;
```

DIY IoT

- Example Programming
  - Simple ESP8266 program with networking

```
void handleRoot() {
  digitalWrite(led, 1);
  server.send(200, "text/plain", "hello from esp8266!");
  digitalWrite(led, 0);
}
```

DIY IoT

- Example Programming
  - Simple ESP8266 program with networking

```
 void handleNotFound(){
 digitalWrite(led, 1);
 String message = "File Not Found\n\n";
 message += "URI: ";
 message += server.uri();
 message += "\n";
 for (uint8_t i=0; i<server.args(); i++){
   message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
 }
 server.send(404, "text/plain", message);
 digitalWrite(led, 0);
}
```

DIY IoT

- Example Programming
  - Simple ESP8266 program with networking

```
void setup(void){
  pinMode(led, OUTPUT);
  digitalWrite(led, 0);
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

DIY IoT

- Example Programming
  - Simple ESP8266 program with networking

```
Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  server.on("/", handleRoot);
  server.on("/inline", [ ](){
    server.send(200, "text/plain", "this works as well");
  });
  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("HTTP server started");
}
```

DIY IoT

- Example Programming
  - Simple ESP8266 program with networking

```
void loop(void){
  server.handleClient();
}
```

DIY IoT

- To Use ESP8266 in Arduino IDE

Under File->Preferences add

http://arduino.esp8266.com/stable/package_esp8266com_index.json

To “Additional Boards managers URLS”

DIY IoT

- Other Resources
  - Arduino IDE - https://www.arduino.cc/en/main/software
  - ESP8266 Arduino reference doc
    - http://esp8266.github.io/Arduino/versions/2.3.0/
  - Mosquitto MQTT broker - https://mosquitto.org/download/
  - Node-RED flow manager - https://nodered.org
  - Youtube videos
    - Node-Red MQTT on Raspberry Pihttps://youtu.be/WxUTYzxIDns
    - Installing mosquitto https://youtu.be/Y-H6grpWdec

DIY IoT

- Other Resources
  - Few books available
- ESP8266: Programming NodeMCU Using Arduino IDE - Get Started With ESP8266 by UpSkill Learning
- Building an IoT Node for less than 15 $: NodeMCU & ESP8266 by Claus Kühnel
- Learning ESP8266 — Build the Internet of Things with the Arduino IDE and Raspberry Pi
  - Not yet released
- Neil Kolban ebook
  - http://neilkolban.com/tech/esp8266/

DIY IoT

- Summary
  - The price of MCU hardware with WIFI / Bluetooth makes this cost effective now
    - Esp8266
    - Esp32
    - omega2
  - The price of MPU (SBC) hardware makes it possible to keep your data contained with only going outside with specific data
    - Raspberry Pi
    - Beaglebone
    - omega2