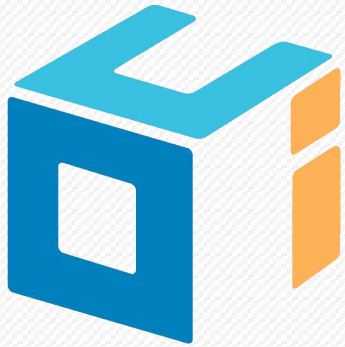
Pronghorn Embedded Toolkit

A simple, effective Java solution for IoT



OCI | WE ARE SOFTWARE ENGINEERS.

Object Computing, Inc.
12140 Woodcrest Exec. Dr., Ste. 250
Saint Louis, MO 63141 USA

What is the Internet of Things (IoT)?

Automatically exchange data over a network

Between physical **devices**, **machines**, and **things**

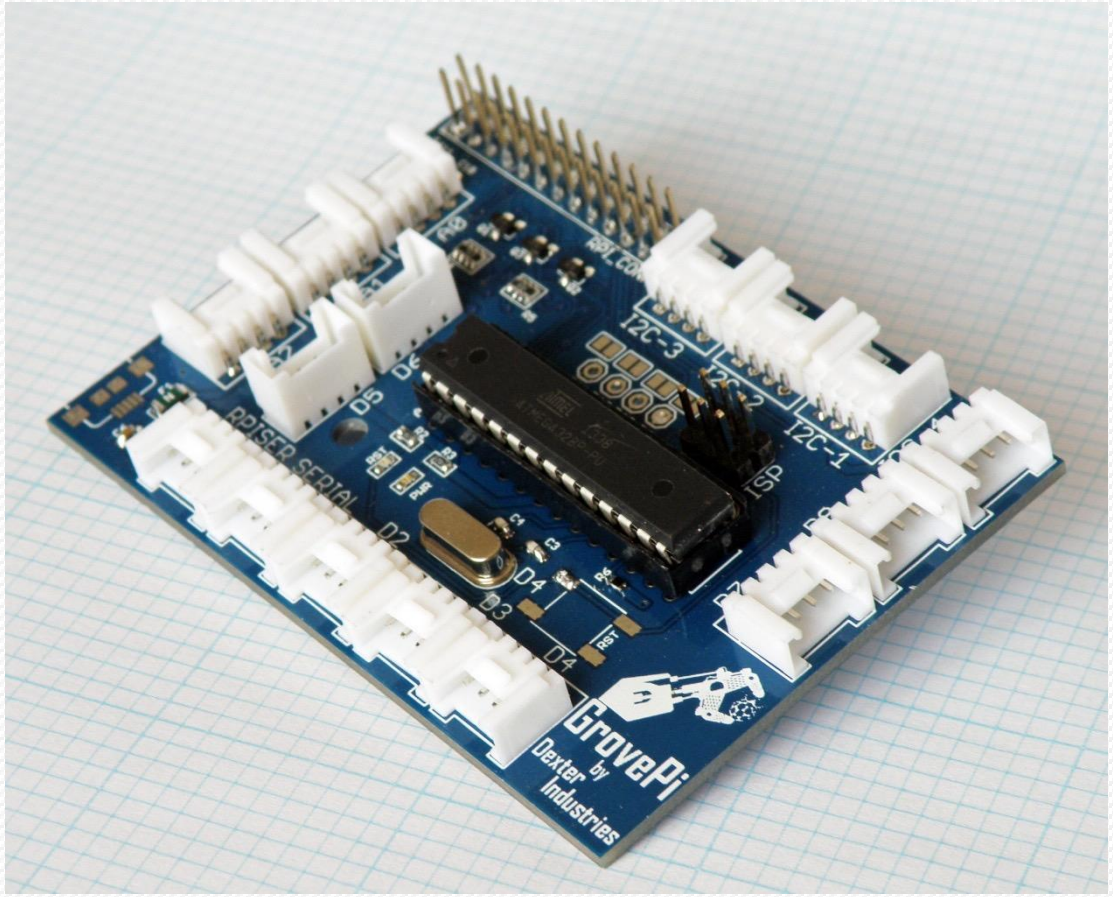
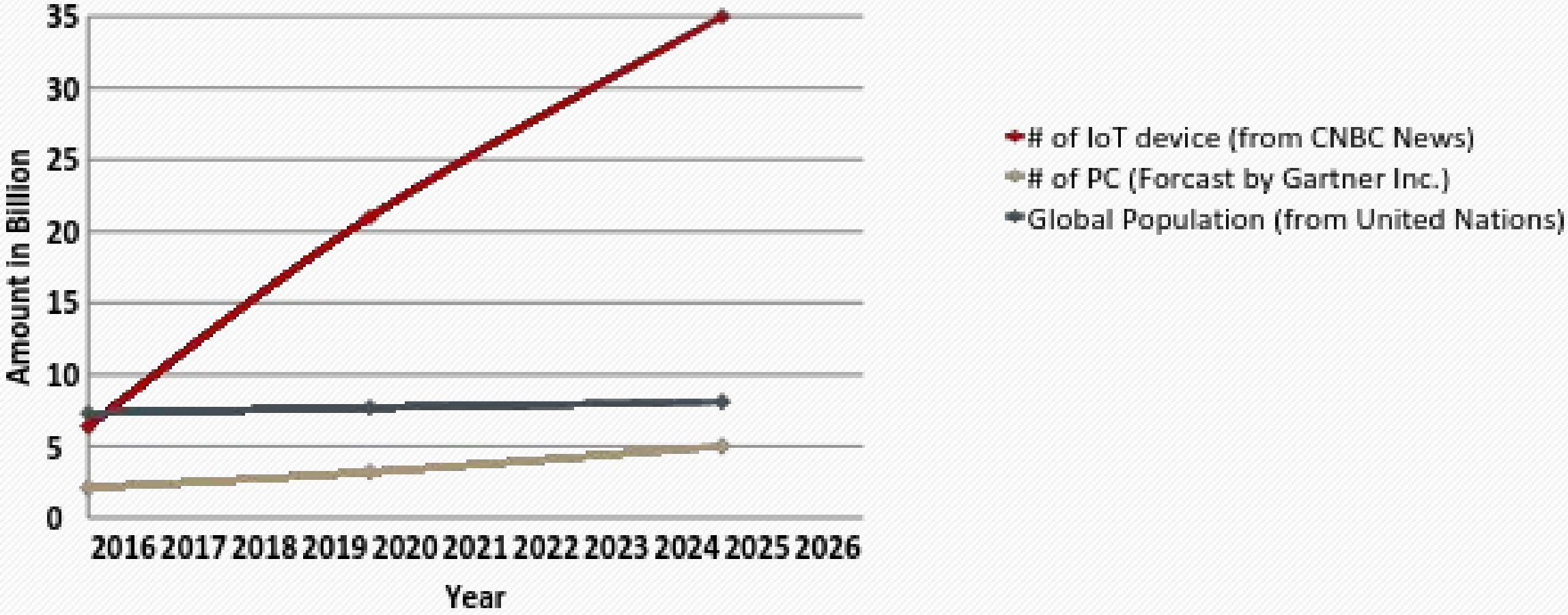


Photo of a Grove-Pi IoT device

Rapid Growth Rate of IoT Devices



IoT Development Needs

- **Security** (data are connected to the Internet)
- **Remote management** of the device and applications
- **Local processing** and analytics of sensor data (filtering)
- **Java APIs** for industry-specific IO protocol stacks

What is Pronghorn Embedded Toolkit?

- Makers: primarily middle/high school students
- Devices: Embedded devices that run **Java Virtual Machine (JVM)**
- Makers learn Java 8 functional programming
- Easy to understand (**declarative programming**)

Low System Requirement

- Low System Requirement:
  - Compact 1 profile comes in less than 16MB
  - Jar is 1.5MB
- Open JDK

A Different Approach

Declarative Programming

- Makers describe what to do but not necessarily how to do it
- Declare Result ("What")
- Easier to Use/Learn
- Ex. Turn on light

Imperative Programming

- Makers provide step-by-step procedure to the compiler
- Declare the procedure ("How")
- Commonly used by programmers
- Ex. Write HIGH to the register

Imperative and Declarative Example

Goal: Find all odd number in List<int> collection = new List<int> { 1, 2, 3, 4, 5 };

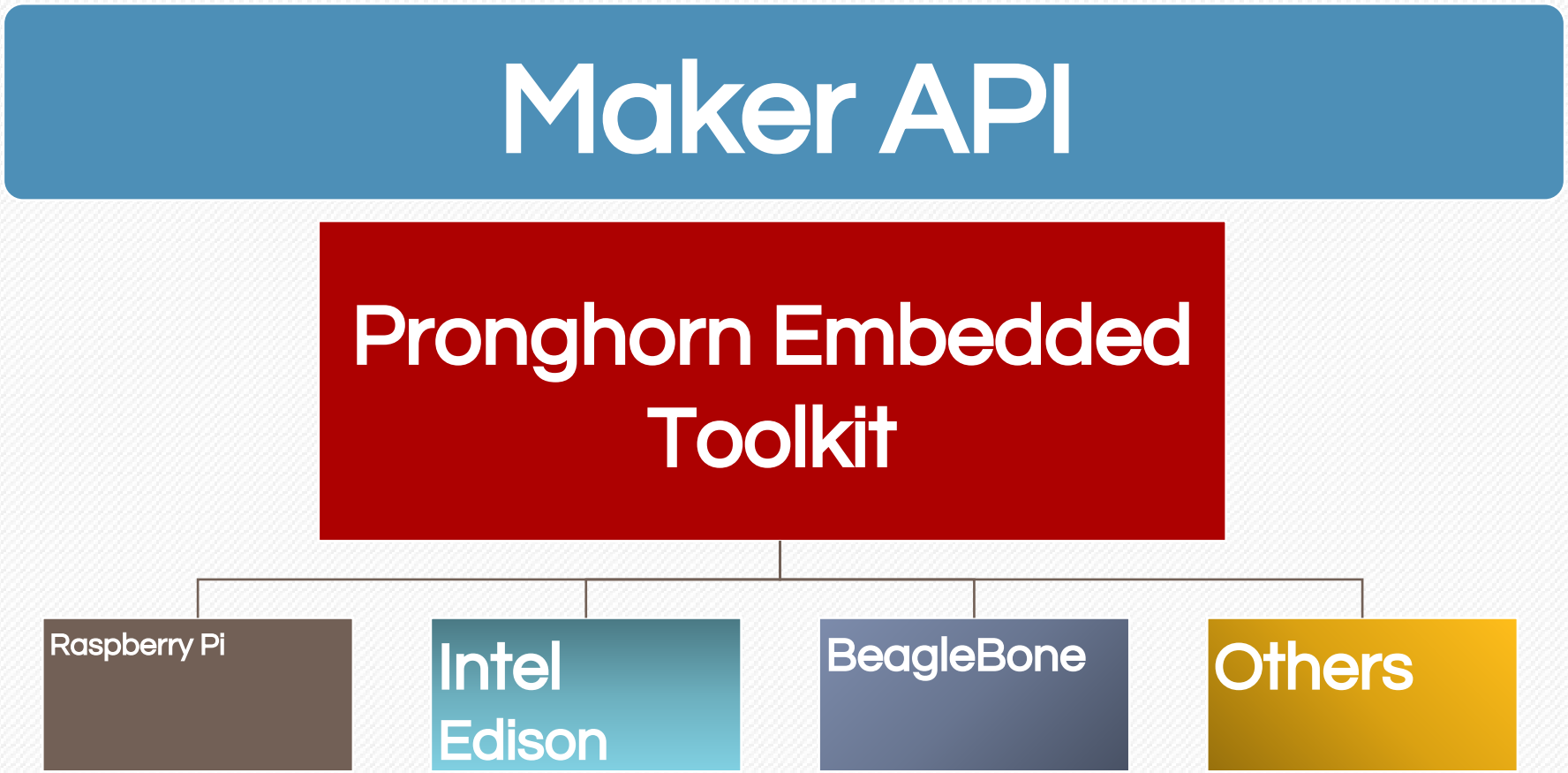
Declarative Approach Example

```
var results = collection.Where( num => num % 2 != 0);
```

Imperative Approach Example

```
List<int> results = new List<int>();

for (int i=0; i<collection.length;i++) {

        if (collection[i] % 2 != 0){

                results.Add(num);

        }

}
```

Leave the "How" Part to Pronghorn



Compare to Arduino

Arduino	Pronghorn Embedded Toolkit
•Single Process Thread	•Multiple Process Thread (Faster for multi-core embedded devices)
•Procedural (C language)/Imperative Programming	•Event driven
	•Functional/Declarative Approach

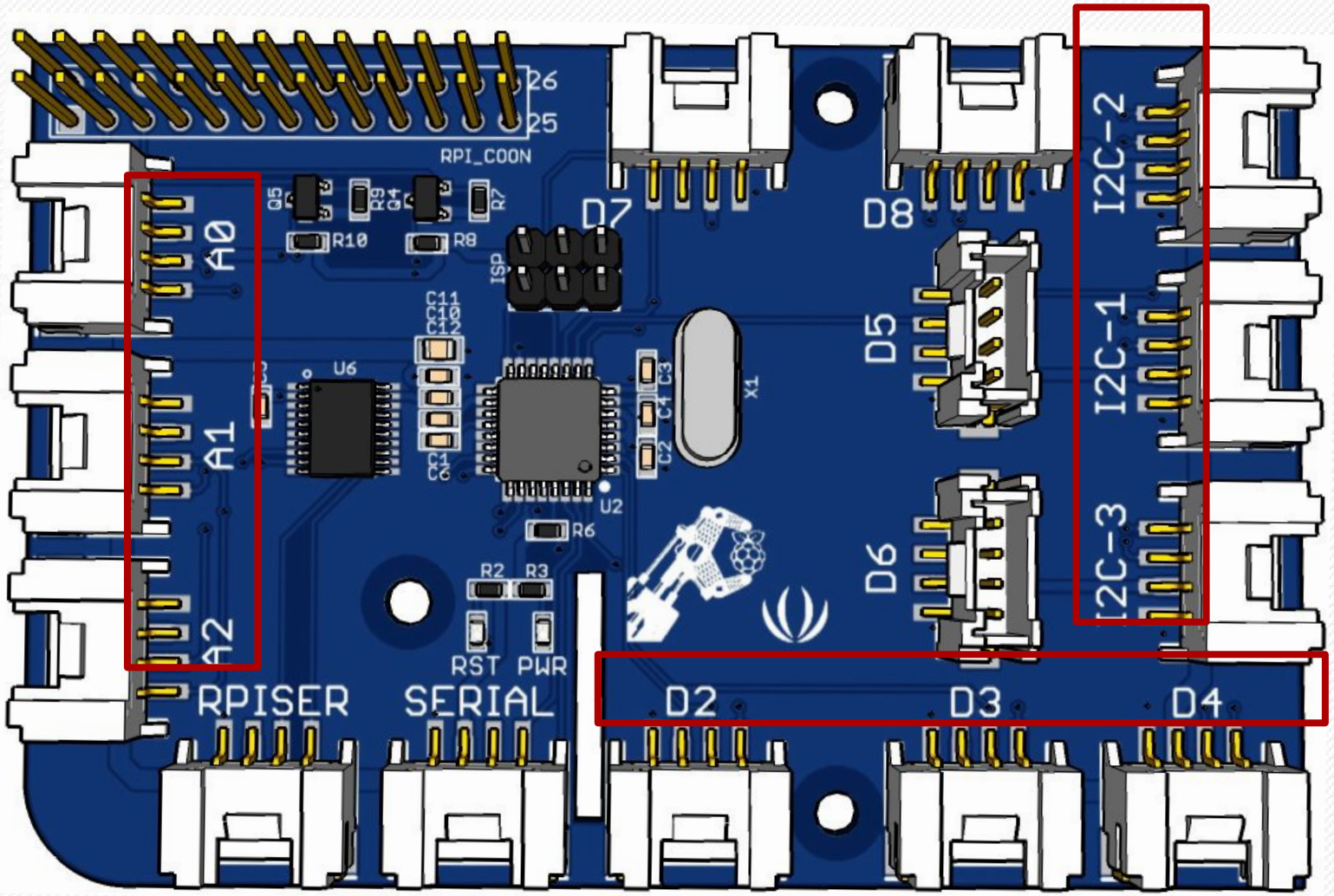
Arduino and Pronghorn

```
int buttonPin1 = 5;
int buttonPin2 = 6;
int ledPin1 = 7;
int ledPin2 = 8;
int buttonDelta = 10;
int lightDelta = 5000;
long lastButton1 = 0;
long lastButton2 = 0;
long lightEnd = 0;

void setup() {
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
}

void loop() {
  if(lastButton1 + buttonDelta > millis()){
    lastButton1 = millis();
    digitalWrite(ledPin1, digitalRead(buttonPin1));
  }
  if(lastButton2 + buttonDelta > millis()){
    lastButton2 = millis();
    lightEnd = millis() + lightDelta;
  }
  digitalWrite(ledPin2, millis()<lightEnd);
}
```

```
public class IoTApp implements IoTSetup

    public static final Port BUTTON1_CONNECTION = D5;
    public static final Port BUTTON2_CONNECTION = D6;
    public static final Port LED1_CONNECTION = D7;
    public static final Port LED2_CONNECTION = D8;

    public static void main( String[] args ) {
        DeviceRuntime.run(new IoTApp());
    }

    @Override
    public void declareConnections(Hardware c) {
        c.connect(Button, BUTTON1_CONNECTION);
        c.connect(Button, BUTTON2_CONNECTION);
        c.connect(LED, LED1_CONNECTION);
        c.connect(LED, LED2_CONNECTION);
        c.useI2C();
    }

    @Override
    public void declareBehavior(DeviceRuntime runtime) {
       CommandChannel channel1 = runtime.newCommandChannel();
       CommandChannel channel2 = runtime.newCommandChannel();
        runtime.addDigitalListener((connection, time, durationMillis, value)->{
            if(connection == BUTTON1_CONNECTION){
                channel1.digitalSetValue(LED1_CONNECTION, value);
            }
            else if(connection == BUTTON2_CONNECTION){
                channel2.digitalPulse(LED2_CONNECTION, 5000);
            }
        });
    }
}
```

Ports on the Board

Analog Input/Output



I²C Input/Output

Digital Input/Output

Photo of a Grove base shield

Connecting to Ports

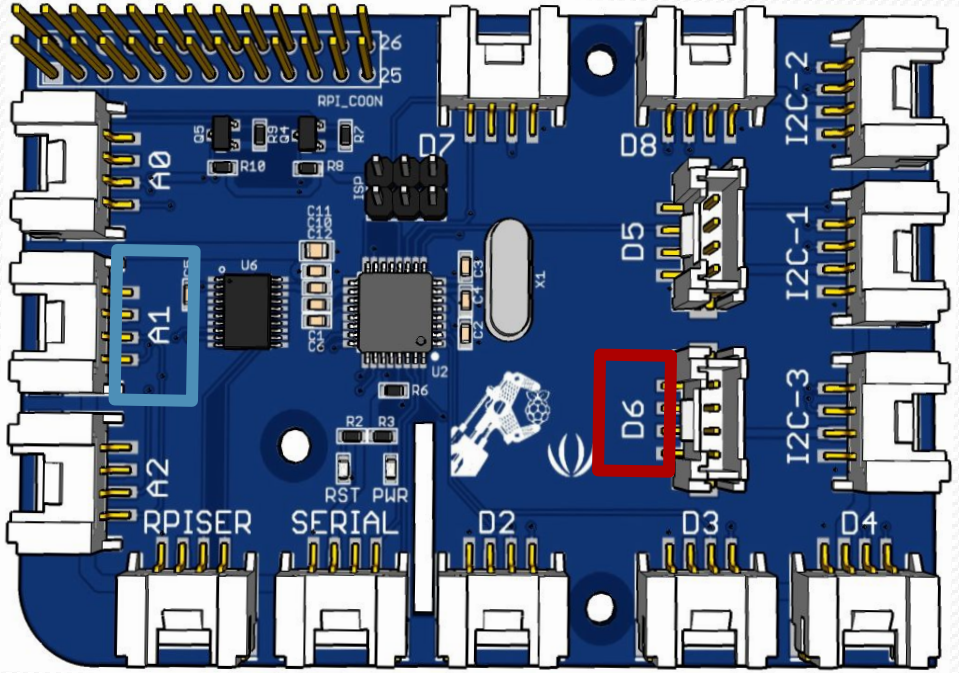
Assume:

- Digital twig (LED)connect to **D6**

  Port connection = **D6**;

- Analog twig (Light Sensor) connect to **A1**

  Port connection = **A1**;



Hello World Sketch in Arduino

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  lcd.print("hello, world!");
}

void loop() {
  lcd.setCursor(0, 1);
}
```

Maker needs to know:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2

Single Process Thread

Hello World Example in Pronghorn

```
public void declareConnections(Hardware c) {
c.usei2C(); }
```

No pin specification needed

```
public void declareConnections (DeviceRuntime runtime) {
 final CommandChannel channel0 =runtime.newCommandChannel();

 runtime.addStartupListener(() -> {
      Grove_LCD_RGB.commandForText(channel0, "Hello World"); });
}

public static void main(String[] args){
 DeviceRuntime.run(new DeviceTestDemo());
}}
```

Multiple Process Thread

Connecting A LCD Could Be Challenging

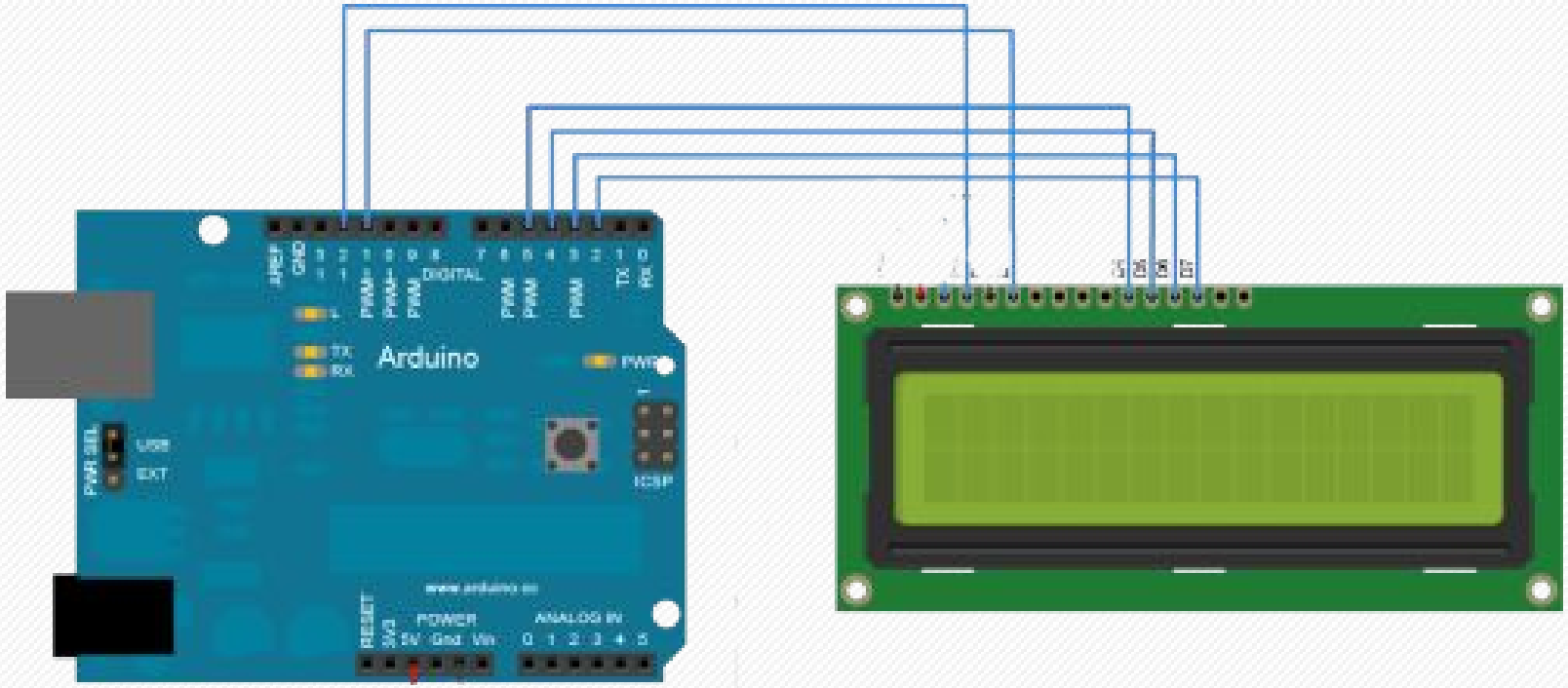


Photo of Simplified Arduino Connection Diagram

General Format

```
public class IoTDeviceDemo implements IoTSetup {

public static void main(String[] args) {

/** execute the behavior declared in the  declareBehavior() method**/

}

public void declareConnections(Hardware c) {

/**Connect kits to the specific ports on Base Shield Board**/

}

public void declareBehavior(IOTDeviceRuntime runtime) {

/**Write instructions to the devices for specific function (Ex. Turn on a LED)**/

} }
```

Look at main()

In the main function

```
public static void main(String[] args) {
    DeviceRuntime.run(new IoTDeviceDemo ()); }
```

- Generic
- Only change: class name (IoTDeviceDemo ())

General Format

```
public class IoTDeviceDemo implements IoTSetup {

public static void main(String[] args) {

/** execute the behavior declared in the  declareBehavior() method**/

}

public void declareConnections(Hardware c) {

/**Connect kits to the specific ports on Base Shield Board**/

}

public void declareBehavior(DeviceRuntime runtime) {

/**Write instructions to the devices for specific function (Ex. Turn on a LED)**/

} }
```

Connection()

Specify Pin Configuration

- The makers don't need to know the startup pin configuration
- For Analog and Digital, makers need to specify:

```
c.connectDigital (LED, D6); // connect a digital twig
c.connectAnalog (LightSensor,A1);// connect an analog twig
```

- For I2C, no pin specification is needed

```
c.useI2C();
```

General Format

```
public class IoTDeviceDemo implements IoTSetup {

public static void main(String[] args) {

/** execute the behavior declared in the  declareBehavior() method**/

}

public void declareConnections(Hardware c) {

/**Connect kits to the specific ports on Base Shield Board**/

}

public void declareBehavior(DeviceRuntime runtime) {

/**Write instructions to the devices for specific function (Ex. Turn on a LED)**/

}}
```

declareBehavior to turn on the LED

Declare to Turn on the LED

- Need to instantiate a channel for analog, digital, or I2C object:

```
final CommandChannel channel0 = runtime.newCommandChannel();
```

- A listener is needed to start or get input from a twig:

```
runtime.addStartupListener(() ->{
        Channel0.digitalSetValue(6,1);});
```

"6" is the connection, "1" is "ON"

The Built-in Timing Framework

- For time sensitive device
- Rate of polling data:
  - Not to miss data
  - Polling frequently is costly
- Makers don’t need to deal with the timing concept

Timing and States

```
runtime.addDigitalListener((connection, time, value)->{

        count +=value;

    state = (count)%NUM_STATES;//define number of states

    System.out.println("the state is:" + state);

}); // this method will create states for a state machine
```

- Introduce the states concept instead of loops

Achieve Concurrent Programming

Command Channel 1	Command Channel 2
Turn on Port 2	Turn on Port 4
Wait 1000 seconds	Wait 400 seconds
Turn off Port 2	Turn off Port 4
…	Turn on Port 2
	…

Take 1000 s

Take 400 s

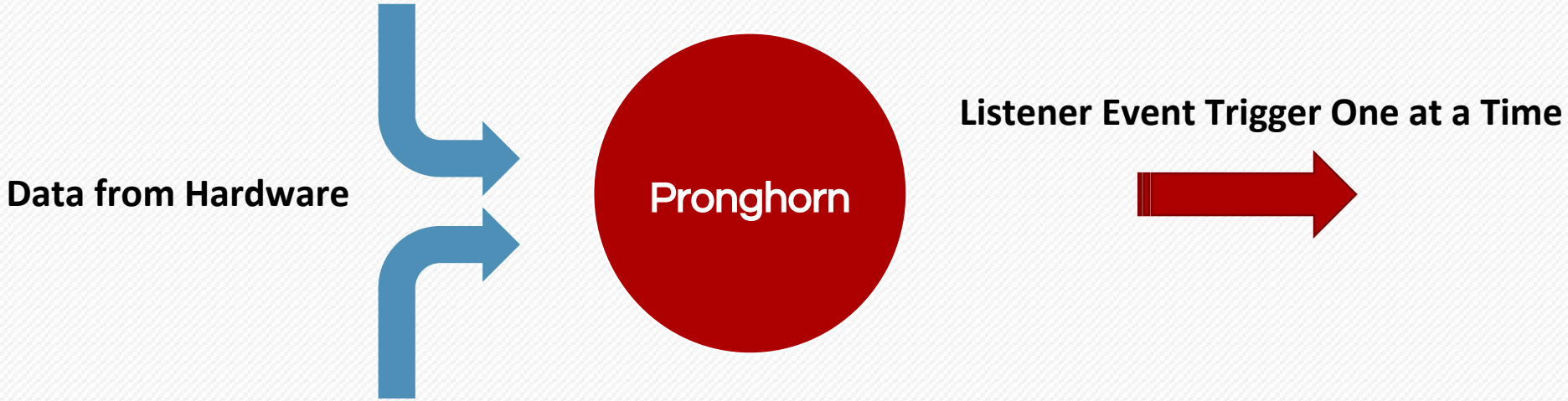
Run Asynchronously

Pronghorn will block Port 2 till CC1 turn it off

Achieve Concurrent Programming

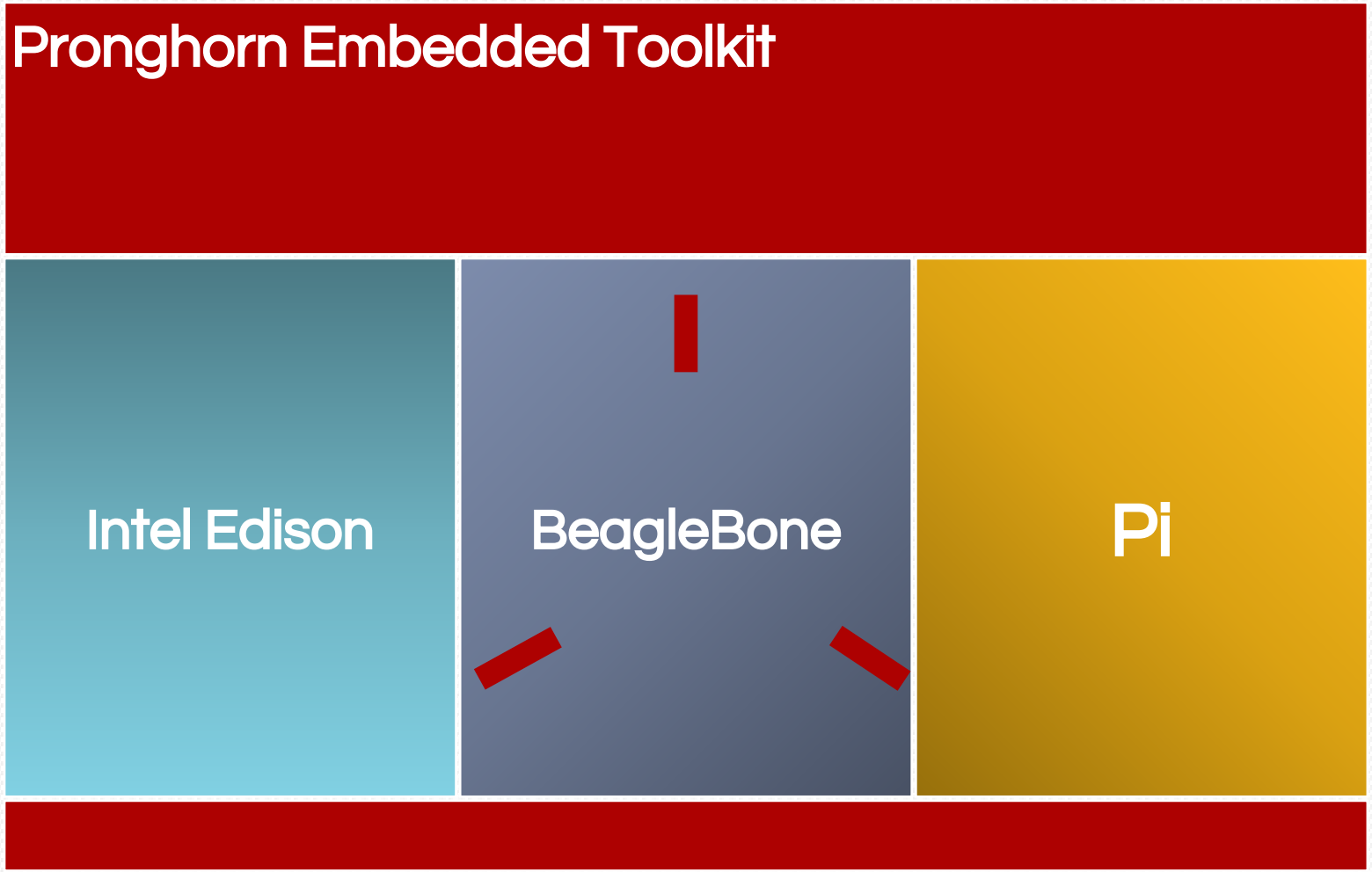
The other half of solving concurrency:

- Responses are back one after another



Pronghorn IoT Approach

Same Code Runs on Many Devices



Pronghorn IoT Approach

Simple Programing

Maker	Pronghorn API
• Specify Twig connection	• Pin configuration
• **Declarative** instructions	• Manage **concurrency**
• Focus on implementing the IoT device	• Abstracts the low-level native integration

Alternatives to Pronghorn

Pi4J

- Java
- Only for Pi
- Low level
- Imperative Programming

LibMraa

- Mostly JavaScript/Python
- For Intel product and Pi
- Low level
- Imperative Programming

Turn on a LED using LibMraa in Java

Try to load the “mraajava” library

```
public class BlinkIO {
    static {
        try {
            System.loadLibrary("mraajava");
        } catch (UnsatisfiedLinkError e) {
            System.err.println(e);
            System.exit(1);
        }
}
```

The Body of the main()

```
public static void main(String argv[]) throws InterruptedException {
        int iopin = 6;
          Gpio gpio = new Gpio(iopin);
            Result result = gpio.dir(Dir.DIR_OUT);
            if (result != Result.SUCCESS) {
              mraa.printError(result);
              System.exit(1);
            }
        gpio.write(1);}}
```

Turn on a LED using LibMraa

(JavaScript)

```
var m = require('mraa');

var myDigitalPin = new m.Gpio(6);

myDigitalPin.dir(m.DIR_OUT);

myDigitalPin.write(1);
```

(Python)

```
import mraa

x = mraa.Gpio(6)

x.dir(mraa.DIR_OUT)

x.write(1)
```

Change Color of a LCD using LibMraa

Python Using LibMraa

```
import mraa

# Change the LCD back light
x = mraa.I2c(0)
x.address(0x62)

# initialise device
x.writeReg(0, 0)
x.writeReg(1, 0)

# sent RGB color data
x.writeReg(0x08, 0xAA)
x.writeReg(0x04, 255)
x.writeReg(0x02, 255)
```

Java Using Pronghorn

```
final CommandChannel channellcd
=runtime.newCommandChannel();


runtime.addStartupListener(() -> {
        Grove_LCD_RGB.commandForColor(     channellcd,
170,255,255); });
```

Why Pronghorn Embedded Toolkit

- Pronghorn is simple
- Makers will learn the low level programming later
- Hide concepts that makers don't need to learn
- Still can access all functions of the Embedded device

The Teaching Goal

- Top down Approach
- Allow students with little CS background to turn on a LED in 15m
- Effective (for more sophisticated programming)

Pronghorn API

API Design

- Lock free
- Garbage free
- Simple interface

Result

- Low overhead
- Low latency
- Low memory usage
- Great responsiveness
- Easy for the users

Demo

Traffic Light

OCI | WE ARE SOFTWARE ENGINEERS

```
public class IoTApp implements IoTSetup
{
        private static final Port LED3_PORT = D3;
        private static final Port LED1_PORT = D7;
        private static final Port LED2_PORT = D8;
        private static long prevTime =0;
        private enum State {
                REDLIGHT(1200), GREENLIGHT(1000),YELLOWLIGHT(200);
                private int deltaTime;
                State(int deltaTime){this.deltaTime=deltaTime;}
                public int getTime(){return deltaTime;}
        };
        private State state = State.YELLOWLIGHT;

        public static void main( String[] args ) {
                DeviceRuntime.run(new IoTApp());
        }

        @Override
        public void declareConnections(Hardware c) {
                c.connect(LED, LED1_PORT);
                c.connect(LED, LED2_PORT);
                c.connect(LED, LED3_PORT);
                c.setTriggerRate(500);
                c.useI2C();
        }

        @Override
        public void declareBehavior(DeviceRuntime runtime) {
                final CommandChannel channel1 = runtime.newCommandChannel();
                final CommandChannel channellcd=runtime.newCommandChannel();
                runtime.addTimeListener((time)->{

                });
        }
}
```

```
switch(state){
case YELLOWLIGHT:
        channel1.setValue(LED1_PORT, 0);
        channel1.setValue(LED2_PORT, 1);
        channel1.setValue(LED3_PORT, 0);
        Grove_LCD_RGB.commandForTextAndColor(channellcd,"YELLOW", 255, 255, 0);
        if(time-prevTime>=state.getTime()){
                state = State.REDLIGHT;
                prevTime = time;
        }
        break;
case REDLIGHT:
        channel1.setValue(LED1_PORT, 1);
        channel1.setValue(LED2_PORT, 0);
        channel1.setValue(LED3_PORT, 0);
        Grove_LCD_RGB.commandForTextAndColor(channellcd, "RED", 255, 0, 0);
        if(time-prevTime>=state.getTime()){
                state = State.GREENLIGHT;
                prevTime = time;
        }
        break;
case GREENLIGHT:
        channel1.setValue(LED1_PORT, 0);
        channel1.setValue(LED2_PORT, 0);
        channel1.setValue(LED3_PORT, 1);
        Grove_LCD_RGB.commandForTextAndColor(channellcd, "GREEN",0, 255, 0);
        if(time-prevTime>=state.getTime()){
                state = State.YELLOWLIGHT;
                prevTime = time;
        }
        break;
}
```

References

Image of “Grove-Pi IoT device”
https://www.gpio.com.au/grove-pi/
Rapid Growth of IoT device
http://www.statista.com/statistics/global-shipments-forecast-for-tablets-laptops-and-desktop-pcs/
http://www.gartner.com/newsroom/
http://www.cnbc.com/2016/02/01/an-internet-of-things-that-will-number-ten-billions.html
http://www.un.org/en/development/desa/news/population/2015-report.html
Image of “Grove base shield board”
http://www.seeedstudio.com/wiki/Grove_-_Base_shield_v2
Image of “Arduino Connection”
http://garagelab.com/profiles/blogs/tutorial-basic-16x2-lcd-with-arduino