Extending Linux with Arduinos

Leveraging the Ecosystem

Mike Anderson
Chief Scientist
The PTR Group, Inc.
http://www.theptrgroup.com

What We Will Talk About…

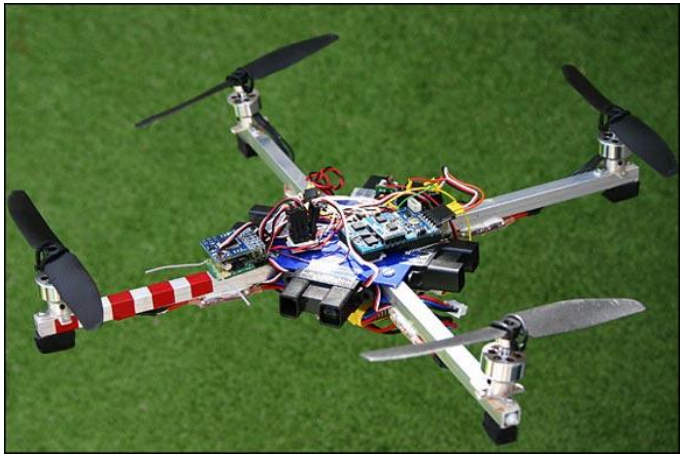
- What is an Arduino?
- Development model
- Why Linux comes up short
- Extending Linux with external μCs
- Connectivity options
- Summary

What is an Arduino?

- Arduino is an open-source SBC based on the Atmel AVR microcontroller line
- The Arduino family uses the Atmel ATmega processor line
  - Purchased from Nordic Semiconductor
    - According to Atmel, AVR doesn't stand for anything
- Using the AVR is like a trip back to the late 1970s
  - 8-bit processor with very limited address space
- Fortunately, the architecture is optimized to execute a HOL like C/C++
  - 32 8-bit registers
  - About 1 MIPs/MHz
- Most processors are either 8MHz or 16MHz
- The goal of the Arduino project was to enable non-technical users with a platform that allowed them to interact with the "real world" in the form of sensors and actuators
  - Originally conceived as a means to create interesting designs and art

Arduino in the Marketplace

- Arduinos are cheap and ubiquitous
  - They range from $8 to as much as $120 depending on options installed
- Arduinos are available from multiple sources including:
  - RadioShack, Frys, MicroCenter, SparkFun and many others
- Applications ranging from simple LED art to fully autonomous multi-rotor sensing and camera platforms with GPS

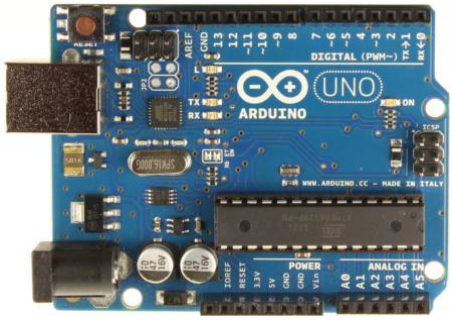


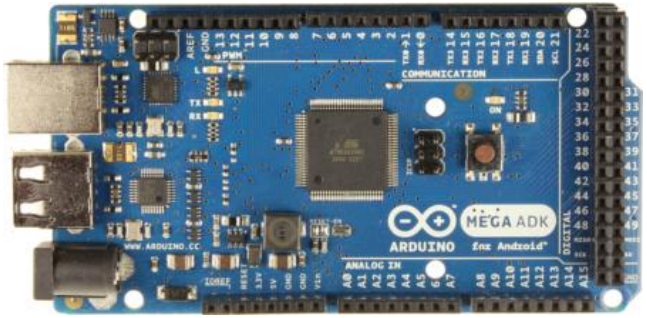
- There is estimated to be over 1,000,000 Arduino and clones in use today

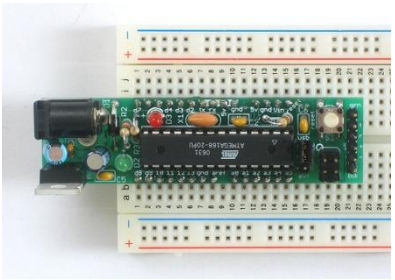
The Arduino Project

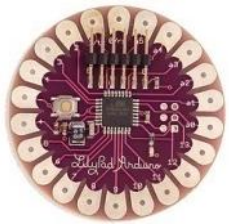
- Ivrea, Italy is the home town of Olivetti
  - Essentially, the Italian version of IBM
- Started in 2005 at the Interaction Design Institute Ivrea in Ivrea, Italy, the Arduino Project's stated mission is:
  - Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.
- The project is forked from the "Wiring" project created by Hernando Barragan as part of his thesis work at Ivrea
- Only "official" boards can carry the Arduino name
  - Uno, Diecimila, Duemilanove, Mega2560, etc.
- However, there are a number of commercially available clones
  - Freeduino, Seeeduino, Boarduino, Netduino, etc.
- Most Arduinos use the megaAVR series of chips
  - ATmega8, ATmega168, ATmega328, ATmega1280 or ATmega2560
    - These have varying amounts of RAM, Flash and I/O
  - Arduino Due uses ARM Cortex M3 (Atmel SAM3x)

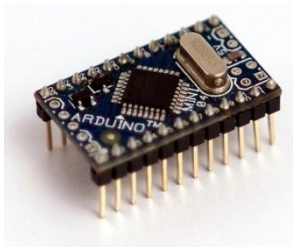
Example Arduinos/Clones

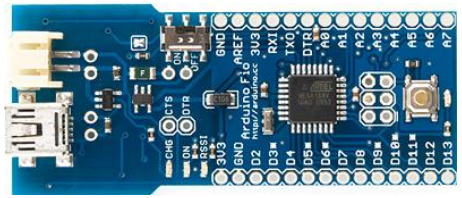


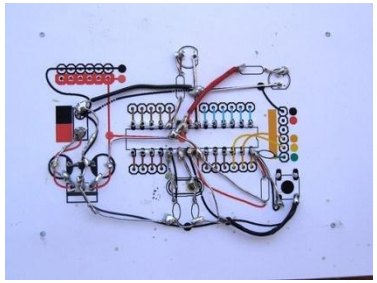


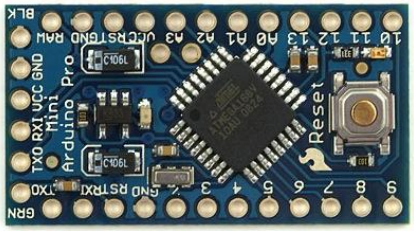


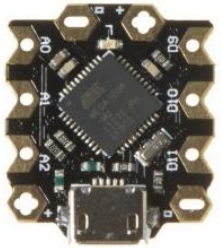












Memory by Processor Type

- This chart shows how much storage you have:

	ATMega168	ATMega328P	ATmega1280	ATmega2560
Flash (1 Kbyte used for bootloader)	16 KBytes	32 KBytes	128 KBytes	256 KBytes
SRAM	1024 bytes	2048 bytes	8 KBytes	8 KBytes
EEPROM	512 bytes	1024 bytes	4 KBytes	4 KBytes

- There are special commands for reading/writing the EEPROM to use as persistent storage of static data such as display strings

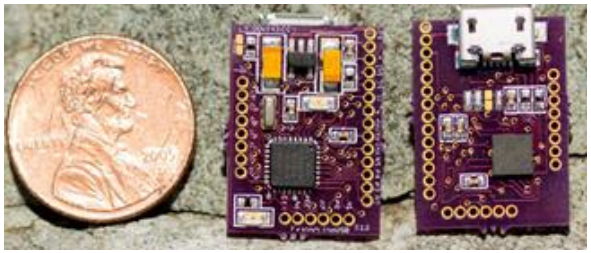
Overview of I/O Capabilities

- The major variants:
  - ATmega328 (Uno)
    - 14 DIO (4 with PWM)
    - 6 analog inputs
    - 2 external interrupt lines
    - 1 UART (simple 3 wire)
    - 2 8-bit, 1 16-bit timer
    - JTAG
  - ATmega2560 (Mega2560/ADK)
    - 54 DIO (14 with PWM)
    - 16 analog inputs
    - 6 external interrupt lines
    - 4 UARTS (simple 3 wire)
    - 2 8-bit, 4 16-bit timers
    - JTAG

- Most Arduinos implement a USB to Serial interface for the UART
  - Used to program the Flash as well as for serial I/O
- Support for I2C, SPI, TWI, UART, A/D, D/A, PWM and GPIOs are all built into the easy-to-use libraries
- There is support for Ethernet via the Wiznet 10/100 Mbps W5100 interface (SPI)
- Wi-Fi, Zigbee and Bluetooth are supported too as is the 423 MHz ISM band
  - RF ranges can be > 2 miles in the lower RF bands

Why Hasn't Linux Killed Arduino?

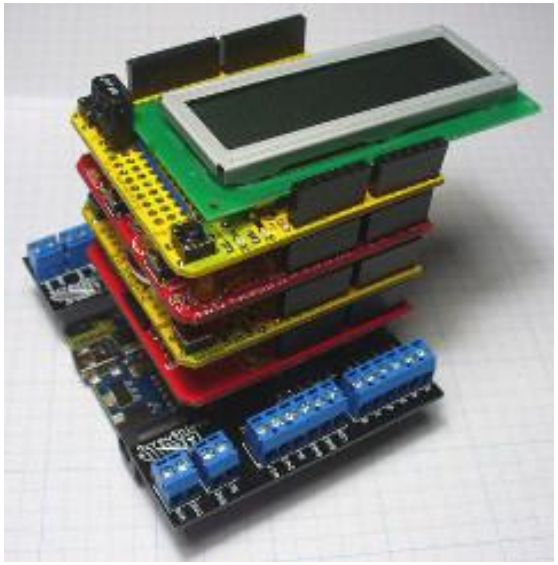
- With the Beaglebone Black, Raspberry Pi, Udoo and others being so cheap, why does Arduino continue to exist?
  - It's not cost
    - Arduinos can cost more than the BBB or RPi
- Size and power are part of it
  - You can buy really tiny Arduino clones
  - You can run an Arduino for months from AA batteries



- Complexity is a big factor
  - Just the process of getting Linux to run on BBB or RPi can be daunting to non-Linux folks
- However, it's the Arduino community and ecosystem that dwarfs what we've accomplished so far with the Linux platforms
  - Arduino programming model is dirt-dumb simple
  - Large selection of libraries available in source code
  - I/O expansion selection is mind boggling
    - The Arduino shield pin-out is almost universal

I/O Shields

- A variety of shields are available:
  - Bluetooth, ZigBee, Ethernet, GPS, protoboard, relays, MIDI, SD Card, LCD, motor controllers, joysticks and many, many more
  - Over 250 shields at last count!
- Some shields can be stacked to create complex systems



The Arduino Boot Cycle

- Ranging from .5 to 1KB, the Arduino bootloader is stored in the Flash
  - Executed on power-on
  - All Arduino boards already have this installed but you can load your own from the IDE
- Runs whatever program is stored in Flash
  - Flash is programmed via JTAG (USB or serial)
- The programming model is very simple
  - No RTOS, just a simple run-time executive and C run time
- No multi-tasking although ISRs are supported
  - Software interrupts can be simulated using the pin change feature

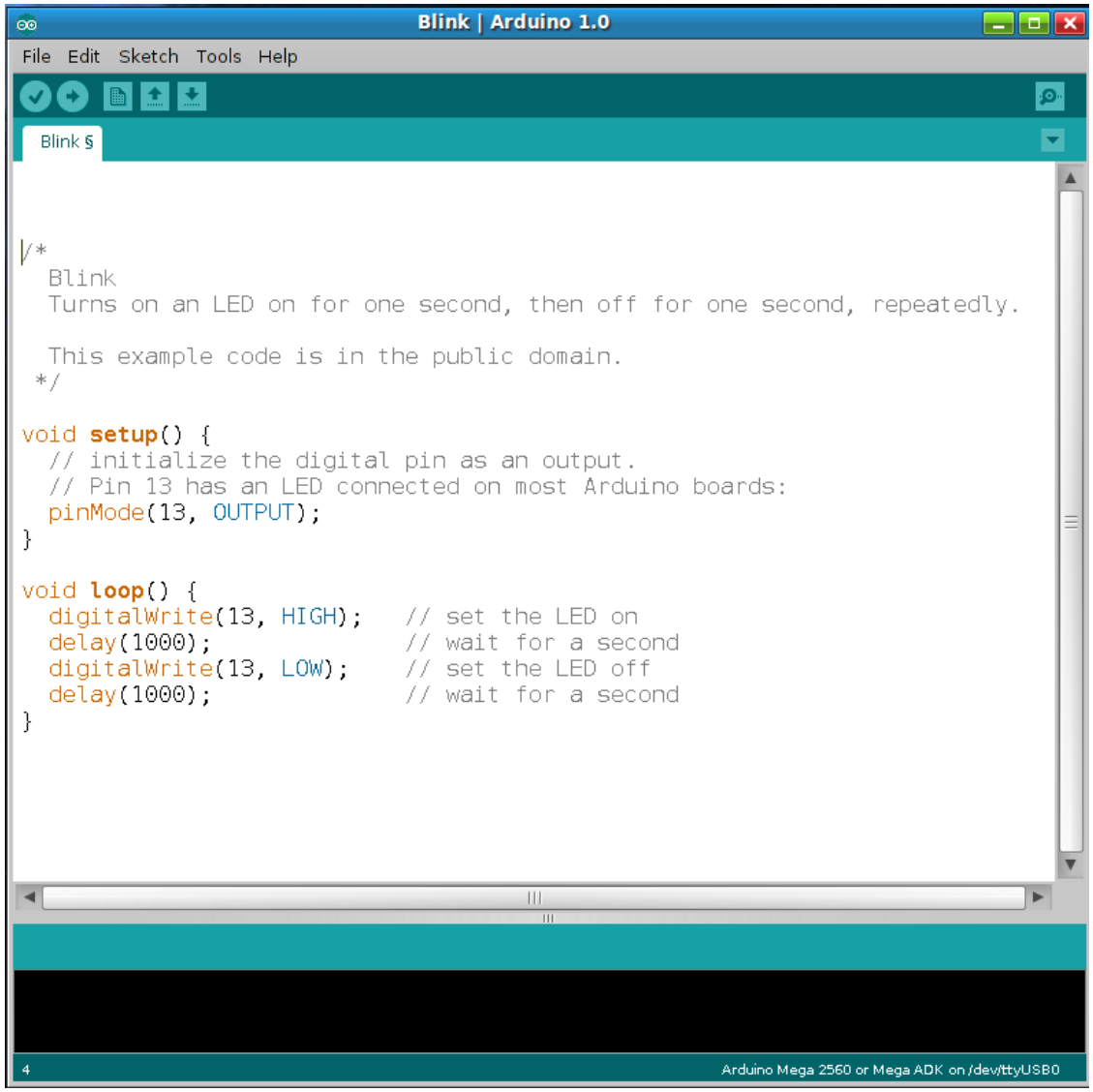
The Arduino Development Environment

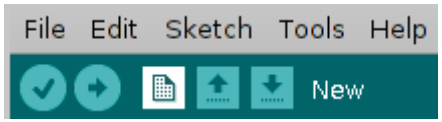
- Just as the hardware is based on the open-source "wiring" model, the programming model is based on the open-source Processing Programming Language
  - Again, targeted at non-professional developers
- A Java-based IDE is available for Windows, OS/X and Linux
  - Open-source and free to download
    - http://arduino.cc/en/Main/Software
  - Implements the basics of syntax highlighting, brace matching and automatic indentation
- The compiler that is included with the IDE is the GNU avr-gcc compiler
- You can also program in AVR assembler
  - But, that kind of defeats the easily accessible part of the Arduino

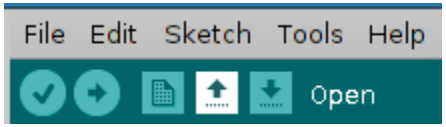
Language Support

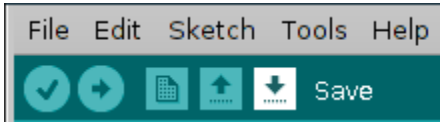
- The Processing language is a restricted subset of C/C++
  - Heavily leverages the use of libraries to accomplish most operations
- Most of C/C++ is supported
  - This includes classes/constructors/destructors, etc.
    - Remember, everything must fit into storage!
- No PVFs, multiple inheritance, RTTI, etc.
  - The stuff that eats memory ;-)
- **`int`** is16-bit but **`long`** is 32-bit
- **`float`** and **`double`** are both 4 bytes
  - Floating point is done in software so consider converting to fixed point to speed computation
- Yes, pointers and dereferences are supported!

Example Arduino IDE









Example Sketch

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

- The Beaglebone's BoneScript tries to approximate this API
- Intel's Galileo board has a version of the Arduino API ported to it
  - But, still runs Linux natively

A simple ISR

- Here is an example that generates an interrupt every time a rising edge is encountered

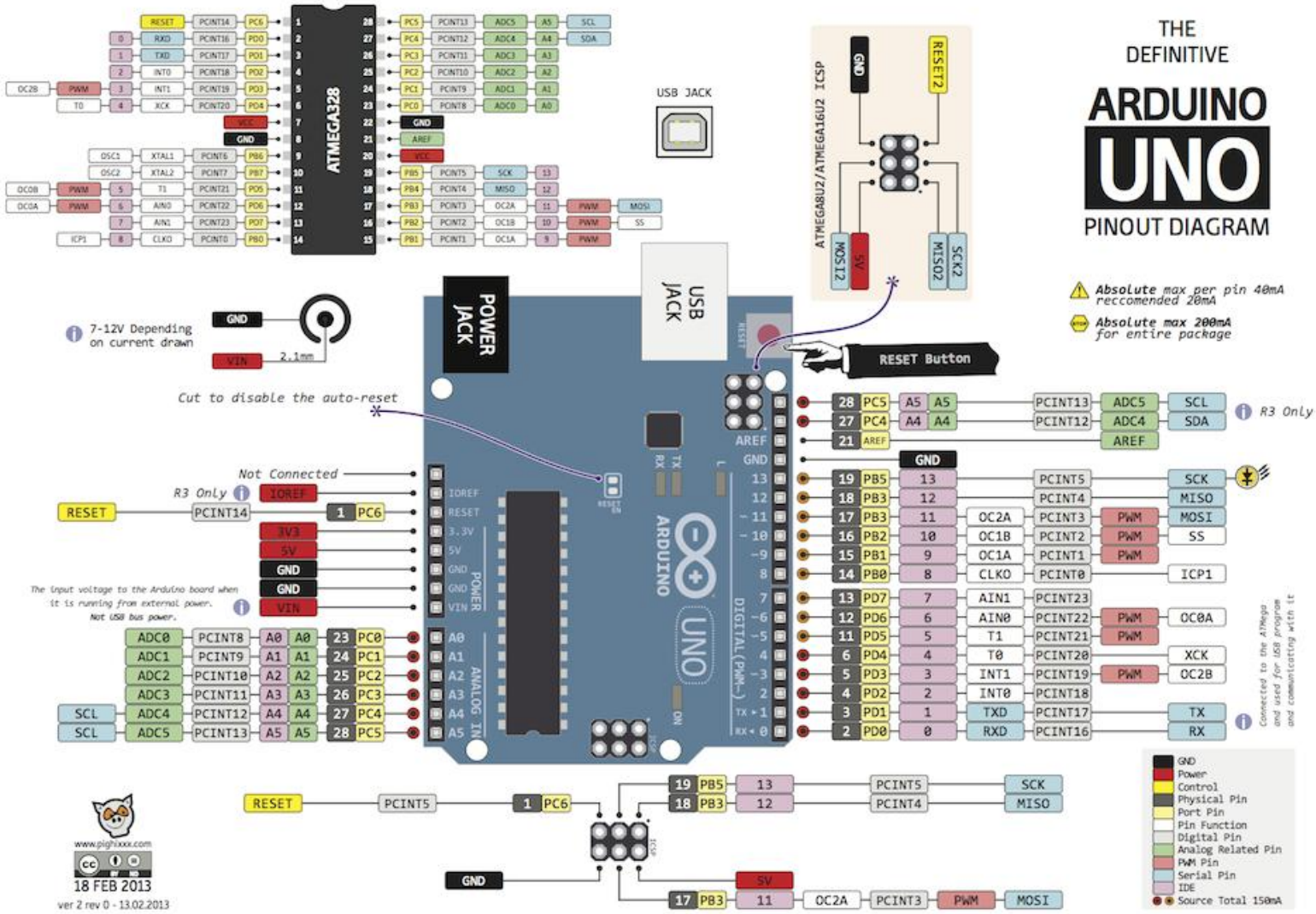
```
void setup(void) {
    attachInterrupt(0, count, RISING);  // Captures external
                                        // interrupt 0
    Serial.begin(9600);                 // Begin Serial at 9600 bps
    Serial.println(" Initializing, Please wait...");

    }


void count() {                      // Function called by AttachInterrupt
                                    // at interrupt at int 0
      siPulseCounter++;             // increment the pulse count

    }
```

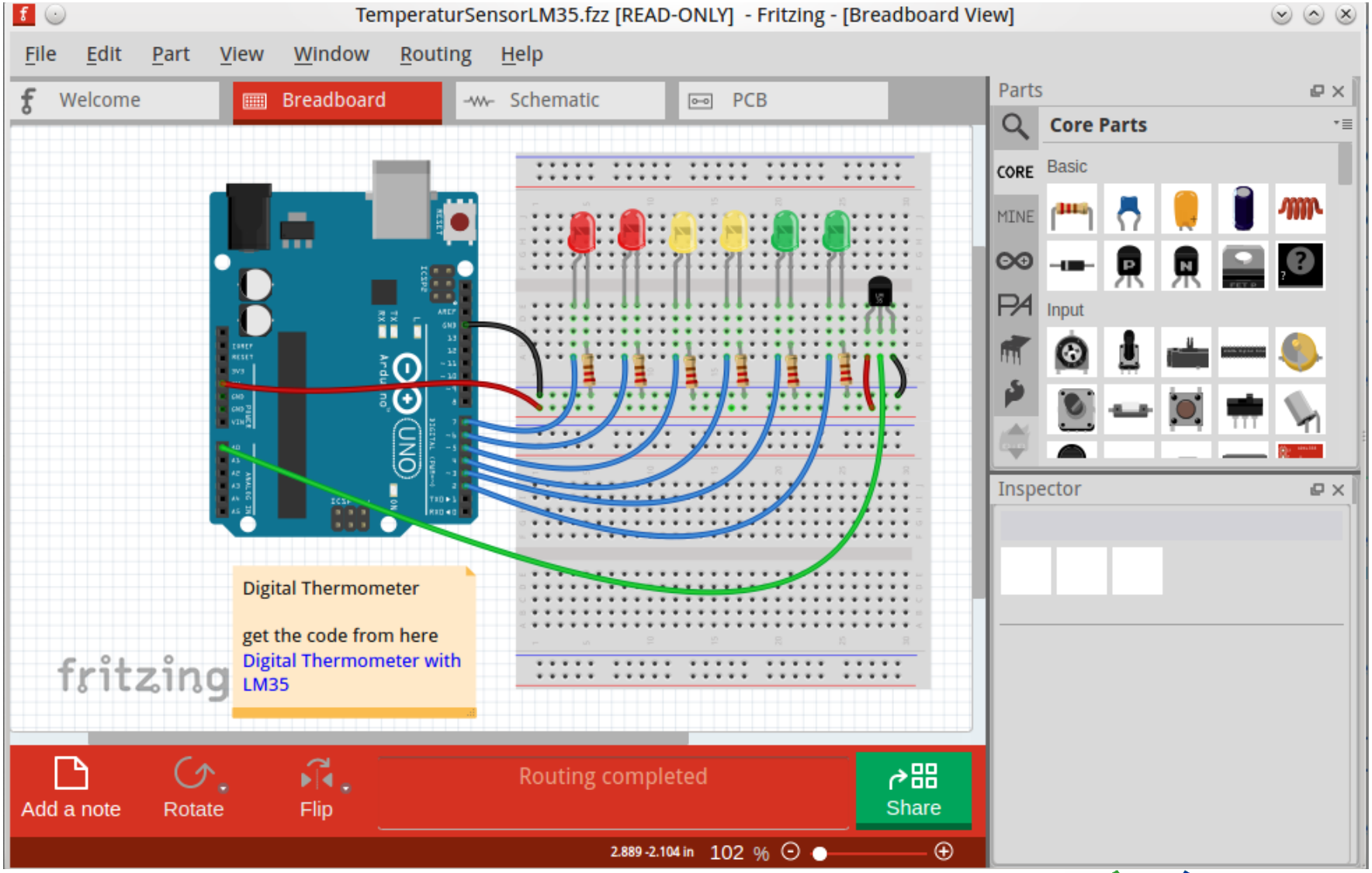
The Arduino Pin-Out



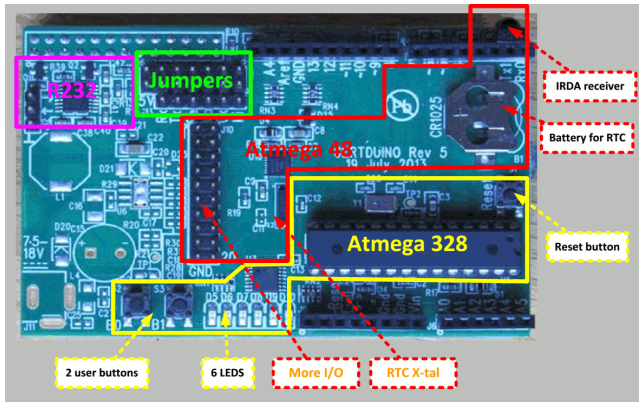
Fritzing Diagrams

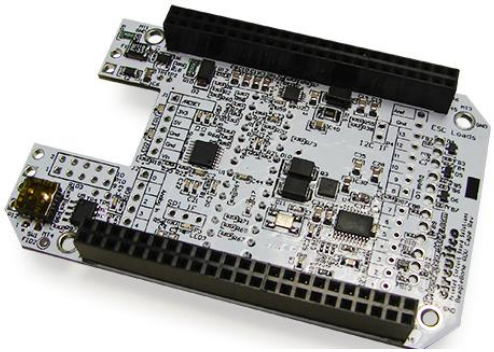
- In order to help non-engineers with wiring prototypes, the Fritzing Diagram approach was created
  - Developed at the University of Applied Sciences of Potsdam
- The software is created in the spirit of Processing and Arduino and allows a designer, artist, researcher, or hobbyist to document their Arduino-based prototype and create a PCB layout for manufacturing
  - Includes the ability to create a schematic as well as the picture of the circuit

Example Fritzing Diagram



Linux/Arduino Boards





Why Extend Linux with Arduinos?

- With PREEMPT_RT, Linux has excellent timing characteristics
  - But, this requires patching the kernel and rebuilding
    - Beyond the grasp of even typical power users
- PREEMPT_VOLUNTARY can't meet the timing requirements of many real-world applications
  - E.g., PWM-based motor controllers
  - This is the default for most Linux distros
- x86 may be fast, but one SMI will kill your R-T performance
- Offloading hard real-time constraints to external processors is often the path of least resistance
  - Arduinos are a good option because they're cheap and easy to use
  - Prototyping real-world I/O using Arduinos is very straightforward
    - Hundreds of examples available on the web

Example: I2C in Linux vs. Arduino

```c
//+ Opening the bus +++++++++++++++++++++++++++++++++++++++++++++++++++
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <linux/i2c-dev.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int file;
char *filename = "/dev/i2c-1";
if ((file = open(filename, O_RDWR)) < 0) {
    /* ERROR HANDLING: you can check errno to see what went wrong */
    perror("Failed to open the i2c bus");
    exit(1);
}
//+ Initiating Comms ++++++++++++++++++++++++++++++++++++++++++++++++++
int addr = 0x48;      // The I2C address of the device
if (ioctl(file, I2C_SLAVE, addr) < 0) {
    printf("Failed to acquire bus access and/or talk to slave.\n");
    /* ERROR HANDLING; you can check errno to see what went wrong */
    exit(1);
}

//+ Reading from device ++++++++++++++++++++++++++++++++++++++++++++++++
unsigned char buf[10] = {0};

for (int i = 0; i<4; i++) {
    // Using I2C Read
    if (read(file,buf,2) != 2) {
        /* ERROR HANDLING: i2c transaction failed */
        printf("Failed to read from the i2c bus: %s.\n", strerror(errno));
            printf("\n\n");
    } else {
        /* Device specific stuff here */
    }
}
```

```c
// This example code is in the public domain.

#include <Wire.h>

void setup()
{
  Wire.begin(4);                // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);           // start serial for output
}

void loop()
{
  delay(100);
}

// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany)
{
  while(1 < Wire.available()) // loop through all but the last
  {
    char c = Wire.read(); // receive byte as a character
    Serial.print(c);      // print the character
  }
  int x = Wire.read();    // receive byte as an integer
  Serial.println(x);      // print the integer
}
```

Connectivity Options

- Since Arduinos support virtually every type of connectivity found in the typical Linux system, it's easy to add them to the mix
  - Wi-Fi, Ethernet, Bluetooth, I2C, SPI, UART, etc.
- My robot from the showcase uses a BBB talking to an Atmel 328 via I2C
- Use the Arduino board for the electrical interface and as a smart buffer
  - Jitter on the Linux side won't impact the control system performance

Summary

- Arduinos are great for quick prototypes
  - Development environment is easy to use
    - Lots of examples on the web
  - Bare-metal approach gives you great repeatability
- Linux can require considerable tweaking to get to the point that you can meet R-T deadlines reliably
- Combining the two environments give you the best of both worlds
  - More and more combination boards are coming out to keep the parts count and costs down