ABOUT ME

Dr. Rafael ‘Tico’ Ballagas
Principal Scientist,
Nokia Research Center
Palo Alto, USA
PHYSICAL PROTOTYPING IN INDUSTRIAL RESEARCH

Family Story Play
Pop goes the cell phone
FAMILY STORY PLAY
POP GOES THE CELL-PHONE
OUTLINE

• Why is prototyping important?
• Arduino
• Digital I/O
• Analog Sensors
• Actuators
• State of the art research
WHAT IS A PROTOTYPE?

• Prototyping structures innovation, collaboration, and creativity in the most successful design studios (Kelley and Littman, 2001)

• Designers use prototypes as physical representations of ideas, effectively externalizing cognition and facilitating a “conversation with materials” to uncover surprising problems or generate suggestions for new designs (Schön and Bennett, 1996).

• Prototypes also serve as artifacts that represent tacit knowledge of developers as a communication tool to clients or other members of a design team (Schrage, 1999).
THE NATURE OF PROTOTYPING

• Problems and solutions co-evolve (Dorst & Cross 2011)

• Subproblems are interconnected (Goel and Pirolli 1992)

• Constraints are often negotiable (Schön 1995)

• Solutions are not right or wrong, only better or worse (Rittle and Webber 1973)
ITERATIVE DESIGN

What is wanted

Interviews
Ethnography

Design

Prototype

Evaluate

Redesign

OK?

Implement and deploy

Yes

No
MORE ITERATIONS = BETTER DESIGNS

(\textit{Nielsen, 1993})

Usability

Reconceptualizing the interface

Removing interaction bugs

Iteration
PARALLEL PROTOTYPING LEADS TO BETTER DESIGN [Dow, 2010]

- Study comparing parallel to serial prototyping
- Parallel prototyping outperformed Serial by all objective performance measures
- Parallel prototypers had more divergent ideas
- Parallel prototypers react more positively toward critique
Design Space of Input Devices

Analysis of the Design Space of Input Devices.

Fig. 4. A broad range of input devices plotted on the taxonomy. Devices previously classified by Foley, Wallace, and Chan [15] and by Baecker and Buxton [4, 7] are indicated by triangles, squares, and hexagons. Hexagons indicate devices included in both previous taxonomies. Other devices, indicated by circles, include the radio devices described previously and some unusual devices to demonstrate the generality of the taxonomy.

To demonstrate the coverage of the taxonomy, we have reclassified the devices listed by Foley, Wallace, and Chan [15] and by Buxton and Baecker [4, 7] (see Figure 4). With the exception of voice, we have been able to position all of the devices considered so far. Furthermore, it is possible to generate potential new devices by placing circles in various cells of the taxonomy.
TRADITIONAL PROTOTYPING

Early Concepts

Form

Interface

Time & Effort

Final Product Concept
PROTOTYPING KITS
# ARDUINO UNO BOARD

http://www.arduino.cc

## Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>ATmega328</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7-12V</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6-20V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>14 (of which 6 provide PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>6</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>40 mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>50 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>32 KB (ATmega328) of which 0.5 KB used by bootloader</td>
</tr>
<tr>
<td>SRAM</td>
<td>2 KB (ATmega328)</td>
</tr>
<tr>
<td>EEPROM</td>
<td>1 KB (ATmega328)</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
</tbody>
</table>
ARDUINO I/O
EXTENDING ARDUINO

- Ethernet shield
- XBee shield
- Motor shield
USB CABLE
9V BATTERY CONNECTOR
BREADBOARDS
BREADBOARDS IN USE
VOLTAGE, CURRENT, RESISTANCE

Ohm’s law

\[ V = I R \]

Voltage Divider

\[ V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in} \]
RESISTORS

Resistor Color Codes: http://www.kpsec.freeuk.com/components/resist.htm
JUMPERS
LIGHT EMITTING DIODE (LED)
RGB LEDS

r1 (180 ohm)
r2 (100 ohm)
r3 (100 ohm)
DIGITAL INPUT & OUTPUT

Switches and LEDs
MY FIRST CIRCUIT

Diagram:

- LED
- Battery: +5V
- Resistor: 220Ω

Handwritten:

Always on
BUTTONS AND SWITCHES
ROCKER SWITCH
LIGHT SWITCH

Diagram:

- LED
- Switch
- Light Switch
- +5V
- 220Ω
LIGHT SWITCH HINT
LIGHT SWITCH SOLUTION
MAKE A LIGHT SWITCH PROGRAMMATICALLY

• Step 1: Programmatically turn on LED
• Step 2: Programmatically sense button press
• Step 3: Programmatically link button to turn on LED
STEP 1: LED

```
int ledPin = 2;
void setup() {
pinMode(ledPin, OUTPUT);
}
void loop() {
digitalWrite(ledPin, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(ledPin, LOW); // set the LED off
delay(1000); // wait for a second
}

Done uploading.
Binary sketch size: 1026 bytes (of a 14336 byte maximum)
```
STEP 2: BUTTON

```c
const int buttonPin = 7; // the number of the pushbutton pin

// variables will change
int buttonState = 0;

void setup() {
    // Create Serial Object
    Serial.begin(9600);
    pinMode(buttonPin, INPUT);
}

void loop() {

    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);

    // check if the pushbutton is pressed.
    // if it is, the buttonState is LOW:
    if (buttonState == LOW) {
        Serial.println("button pressed");
    }
}

Done Saving.

Binary sketch size: 2456 bytes (of a 14336 byte maximum)
```
STEP 3: MAPPING I/O

```cpp
const int ledPin = 2;
const int buttonPin = 7; // the number of the pushbutton pin

// variables will change
int buttonState = 0;

void setup() {
    // Create Serial Object
    Serial.begin(9600);
    pinMode(buttonPin, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);

    // check if the pushbutton is pressed.
    // if it is, the buttonState is LOW:
    if (buttonState == LOW) {
        Serial.println("button pressed");
        digitalWrite(ledPin, LOW);
    } else {
        digitalWrite(ledPin, HIGH);
    }
}
```

Done uploading.
• Download from: http://processing.org

• Processing is an open source programming language and environment for people who want to create images, animations, and interactions

• developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context
PROCESSING EXAMPLE

```java
void setup(){
  size(200,200);
}

void draw(){
  if (mousePressed == true) {
    background(0,0,255);
  } else {
    background(0,0,0);
  }
}
```
ARDUINO & PROCESSING

• Communication over serial port

• Step 1: Arduino to Processing

• Step 2: Processing to Arduino
const int buttonPin = 7; // the number of the pushbutton pin

// variables will change
int buttonState = 0;
int lastButtonState = 0;

void setup() {
  // Create Serial Object
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  if (buttonState != lastButtonState) {
    // check if the pushbutton is pressed.
    // if it is, the buttonState is LOW:
    if (buttonState == LOW) {
      Serial.println(255);
    } else {
      Serial.println(0);
    }
  }
  lastButtonState = buttonState;
}
ARDUINO TO PROCESSING

```java
import processing.serial.*;

float brightness = 0;
Serial port; // The serial port object

void setup(){
    size(200,200);
    // List all the available serial ports
    println(Serial.list());
    port = new Serial(this, Serial.list()[0], 9600);
    port.bufferUntil('\n');
}

void draw(){
    background(0,0,brightness);
}

void serialEvent(Serial port ){
    brightness = float(port.readStringUntil('\n'));
}
```
PROCESSING TO ARDUINO

const int ledPin = 2;
const int buttonPin = 7;  // the number of the pushbutton pin

// variables will change
int buttonState = 0;
int lastButtonState = 0;

void setup() {
    // Create Serial Object
    Serial.begin(9600);
    pinMode(buttonPin, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);

    if (Serial.available() != 0) {
        int val = Serial.read();
        if (val == 0) {
            digitalWrite(ledPin, LOW);
        } else if (val == 1) {
            digitalWrite(ledPin, HIGH);
        }
    }

    // check if the pushbutton has changed
    if (lastButtonState != buttonState) {
        // if it is, the buttonState is LOW:
        if (buttonState == LOW) {
            Serial.println(255);
        } else {
            Serial.println(0);
        }
    }
    lastButtonState = buttonState;
}
import processing.serial.*;

float brightness = 0;
Serial port; // The serial port object
boolean lastMousePressed = false;

void setup(){
    size(200,200);
    // List all the available serial ports
    println(Serial.list());
    port = new Serial(this, Serial.list()[0], 9600);
    port.bufferUntil('\n');
}

void draw(){
    background(0,0,brightness);
    if(lastMousePressed != mousePressed){
        if(mousePressed == true){
            port.write(0);
        } else{
            port.write(1);
        }
    } lastMousePressed = mousePressed;
}

void serialEvent(Serial port){
    brightness = float(port.readStringUntil('\n'));
}
PROCESSING SUPPORT FOR ARDUINO

• http://www.arduino.cc/playground/Interfacing/Processing

• Don’t forget to restart processing after installing the libraries
FIRMATA

• Generic protocol for communicating with microcontrollers from software on a host computer.

• It is intended to work with any host computer software package.

• Works with a number of languages.

• The aim is to allow people to completely control the Arduino from software on the host computer.
LOAD FIRMATA FIRMWARE

- Arduino: File > Examples > Firmata > StandardFirmata
FIRST FIRMATA PROGRAM

```java
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin=2;

void setup()
{
    size(256, 256);
    arduino = new Arduino(this, Arduino.list()[1], 57600);
    arduino.pinMode(ledPin, Arduino.OUTPUT);
    background(0);
}

void draw()
{
    if(mousePressed == true){
        arduino.digitalWrite(ledPin, Arduino.LOW);
    } else {
        arduino.digitalWrite(ledPin, Arduino.HIGH);
    }
}
```
ANALOG SENSORS
POTENTIOMETERS
BEND SENSOR
INFRARED DISTANCE RANGER
2-AXIS ACCELEROMETER
FORCE SENSITIVE RESISTORS
THUMB JOYSTICK
MICROPHONE
PRINCIPLES OF ANALOG INPUT

• 10 bit Analog to Digital Converter

• $2^{10} = 1024$

• Sampling rate up to $\sim 10 \text{ kHz}$

• $V = 5 \times s / 1024$
**/  *
* Graph sensor values  *
*/
import processing.serial.*;
import cc.arduino.*;

int[] xvals;
int arrayindex = 0;
Arduino arduino;
int ledPin=9;
int potPin=0;
void setup()
{
    size(256, 256);
    xvals = new int[width];
    arduino = new Arduino(this, Arduino.list()[1], 57600);
}

void draw()
{
    background(0);
    //shift array left by one
    for(int i=1; i<width; i++) {
        xvals[i-1] = xvals[i];
    }
    // Add the new values to the end of the array
    // read potentiometer (0..1024), divide by four (0..255)
    // to stay within canvas drawing limits
    xvals[width-1] = arduino.analogRead(potPin)/4;
    for(int i=1; i<width; i++) {
        stroke(255);
        point(i, 255-xvals[i]);
    }
}
THRESHOLDING

• process of turning continuous data into discrete yes / no decision
BEND SENSOR

Bend Sensor
10k - 40k

??

Wednesday, July 6, 2011
CALCULATIONS

\[ V_{\text{out}} = \frac{R_2}{R_1 + R_2} \cdot V_n \]

\[ V_n = 5V \]

\[ R_2 = 22k\,\Omega \]

\[ R_1: 10k - 40k\,\Omega \]

**WE EXPECT:**

\[ V_{\text{expected}} = \frac{22k\,\Omega}{(10+22)k\,\Omega} \cdot 5V = 3.44V \]

\[ V_{\text{expected}} = \frac{22k\,\Omega}{40+22k\,\Omega} \cdot 5V = 1.77V \]
EXAMPLE: FSR
THUMB WRESTLING
ACTUATORS
Making things move
SERVO MOTORS
PULSE WIDTH MODULATION

0% Duty Cycle – analogWrite(0)

25% Duty Cycle – analogWrite(64)

50% Duty Cycle – analogWrite(127)

75% Duty Cycle – analogWrite(191)

100% Duty Cycle – analogWrite(255)
EXAMPLE: DIMMER

LED CIRCUIT

- 5V
- 22Ω
- LED
- ARDUINO PIN #9

ANALOG INPUT CIRCUIT

- 0.5V
- POTENTIOMETER
- ARDUINO ANALOG IN #0
- GND
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int pos=0; // servo position in degrees (0..180)
int servoPin=9;

void setup() {
  size(360, 200);
  arduino = new Arduino(this, Arduino.list()[1], 57600); // your
  arduino.pinMode(servoPin, Arduino.OUTPUT);
}

void draw() {
  // read mouseX coordinate
  int newPos = constrain(mouseX/2, 0, 180);
  // update bg & servo position if mouseX changed
  if(newPos != pos) {
    background(newPos);
    arduino.analogWrite(servoPin, newPos);
    pos=newPos; // update servo position storage variable
  }
}
DESIGN EXERCISE
RESEARCH IN PHYSICAL PROTOTYPING
KEY RESEARCH GOAL

• Low Threshold

• High Ceiling
PHIDGETS
www.phidgets.com
VOODOO I/O

Two-pin design

One-pin design

Conductive pin surface

Isolated pin surface

Conductive surface layers

Isolative surface layer
LILYPAD ARDUINO

[Buechley, CHI '08]
D.TOOLS: STATE CHART EDITING
ISTUFF MOBILE: PIPE & FILTER

Live Sensor Value: 225!
EXEMPLAR: PROG. BY DEMONSTRATION
MODKIT

- Scratch Meets Arduino
MODKIT EXAMPLE
TEAR DROP

[Buechley, TEI '09]
TAKEAWAYS

• Build lots of prototypes, both parallel and iterative.

• Key is to fail early and often

• Good prototyping tools provide a low threshold and a high ceiling.