AS220 Workshop

Part II – Interactive Design with advanced Transducers

Lutz Hamel
hamel@cs.uri.edu
www.cs.uri.edu/~hamel/as220
Arduino

How we see the computer

Arduino

How the computer sees us

Principles of Interactive Design

- The “Conversation Loop”
  - Listening (Sensing)
  - Thinking (Processing)
  - Speaking (Acting)
Principles of Interactive Design

Examples of the “Loop”

- Sensing
  - read photoresistor value
  - read potentiometer value

- Processing
  - convert to sound wave period
  - compute PWM duty cycle

- Acting
  - output sound wave to speaker
  - output PWM signal to LED
Arduino Principles of Interactive Design

- Sensing
  - Intensity
    - peak detection
    - threshold setting
  - Duration
    - edge detection

Note: Pulse edge detection is achieved on the Arduino with the 'pulseIn(pin, value)' function.
Principles of Interactive Design

- Sensing
  - Presence
    - foot switches
    - beam breaking
    - motion detection
    - ultrasonic sensors
  - Attention
    - pushbuttons
    - potentiometers
    - flexsensors
Arduino

Principles of Interactive Design

Video Documentation of Blendie (2004)

© Kelly Dobson

Expressive Interfaces
Principles of Interactive Design

Expressive Interfaces

Image source: "turn signal biking jacket", Leah Buechley
Principles of Interactive Design

Expressive Interfaces

Image source: "turn signal biking jacket", Leah Buechley
Acting - Motors

- The actuators we have seen so far provided us with information.
- Motors allow us to act on the world in a more immediate fashion by creating movement.
- Motors come in many different sizes and functionalities, we discuss two types of motors: RC servos and DC motors.
RC Servo Motors

- Remote Control (RC) Motors are DC motors that have control circuitry built-in.
- These motors can be controlled in such a way that you can rotate the spindle between 0° and 180°.
- You guessed it, we use pulse widths to do so.
RC Servo Motors

- A pulse every 20msec
- Pulse width of 1msec $\Rightarrow 0^\circ$
- Pulse width of 2msec $\Rightarrow 180^\circ$
- Good News: we have a library that does that for us!
Idea:

- Write a sketch that sweeps the motor back and forth through its 180 degrees of rotation.
Notes:
- You will need an external power supply.
- The Arduino servo library only supports servos on D9 and D10 (see documentation)
// Sweep
// by BARRAGAN <http://barraganstudio.com>

#include <Servo.h>

Servo myservo; // create servo object to control a servo
    // a maximum of eight servo objects can be created

int pos = 0; // variable to store the servo position

void setup()
{
    myservo.attach(9); // attaches the servo on digital pin 9 to the servo object
}

void loop()
{
    for(pos = 0; pos < 180; pos += 1) // goes from 0 degrees to 180 degrees
    {
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the position
    }
    for(pos = 180; pos>=1; pos-=1) // goes from 180 degrees to 0 degrees
    {
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the position
    }
}
Idea: Use a flex sensor to control the rotation of the servo.

Image source: ITP/NYU
Arduino

RC Servo Motors
// Controlling a servo position using a flex sensor
// based on a sketch by Michal Rinott

#include <Servo.h>

Servo myservo;

int flexpin = 0;
int val;

void setup()
{
  myservo.attach(9);
}

void loop()
{
  // reads value between 0 and 1023
  val = analogRead(flexpin);
  // scale it to a value between 0 and 180
  val = map(val, 500, 400, 0, 180);
  myservo.write(val);
  delay(15);
}
DC Motors

- A DC motor works by converting electric power into mechanical work.
- This is accomplished by forcing current through a coil and producing a magnetic field that spins the motor.
- Able to run forward and backwards, depending on the orientation of the voltage source.
DC Motors

- **Idea:**
  - Control the speed of the rotation of a DC motor using PWM
  - We read an analog signal from a pot and output the appropriate PWM signal to the motor

- **Caveat:**
  - DC motors draw too much current to directly hook up to an IO port
  - Use *transistor* circuitry to drive the motor
Transistors act like switches:

- $I_B \ll I_C$
- If $I_B = 0$ then $I_C = 0$
- If $I_B \neq 0$ then $I_C \neq 0$
Arduino

DC Motors

[Diagram of circuit with Arduino and DC motors]

[Diagram of circuit components]

[Diagram of transistor TO-220 with labels: 1. Base, 2. Collector, 3. Emitter]
// PWM control of a DC motor

int motorPin = 9;  // motor connected to digital pin 9 (PWM)
int potPin = 0;     // pot connected to analog pin 0
int val = 0;

void setup()
{
  pinMode(motorPin, OUTPUT);  // sets the pin as output
}

void loop()
{
  val = analogRead(potPin);
  val = map(val, 0, 1023, 0, 254);
  analogWrite(motorPin, val);
  delay(100);
}
Idea:

- Use a digital control signal to let a motor spin forwards and backwards
- We use an *H-bridge* to accomplish this
- An H-bridge is an integrated circuit (IC) that is able to interpret digital commands and spin the motor in the appropriate direction
Arduino

H-Bridge

SN754410

Motor spins in one direction

Motor spins in the other direction

Command Interpretation

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A High</td>
<td>1Y High</td>
</tr>
<tr>
<td>1A Low</td>
<td>1Y Low</td>
</tr>
<tr>
<td>2A High</td>
<td>2Y Low</td>
</tr>
<tr>
<td>2A Low</td>
<td>2Y High</td>
</tr>
</tbody>
</table>
Arduino

H-Bridge
// digital directional control of a DC motor
// using an H-bridge

int inputPin = 7;    // read digital pin 7
int outputPin1 = 8;  // output pin 1a to bridge digital pin 8
int outputPin2 = 9;  // output pin 2a to bridge digital pin 9
int val = 0;

void setup()
{
    pinMode(inputPin, INPUT);
    pinMode(outputPin1, OUTPUT);
    pinMode(outputPin2, OUTPUT);
}

void loop()
{
    val = digitalRead(inputPin);
    if (val == HIGH) {
        digitalWrite(outputPin1, HIGH);
        digitalWrite(outputPin2, LOW);
    } else {
        digitalWrite(outputPin1, LOW);
        digitalWrite(outputPin2, HIGH);
    }
    delay(100);  // wait 100ms
Driving AC Loads

- We can drive 120V AC loads using relays.
- Relays are mechanical switches that can be switched by applying power to a secondary circuit.

**WARNING ☠:** 120V AC current can kill! If you have not done this before consult with somebody who has experience!
Building Your Objects

- A great way to explore the mechanical side of your objects are construction sets like
  - LEGO
  - Erector/Meccano sets
- Another way is to hack old toys
- Ebay is a great place to shop for these things
Arduino

Things to do for next Week

- Design your interactive Object(s)
  - Briefly describe the behavior
    - input/output
    - “Conversation Loop”
  - Sketch your overall design
  - Layout the hardware
  - Design the Arduino Sketch
- Bring this all to the next Workshop