CSCI1600: Embedded and Real Time Software

Lecture 5: Arduino Programming
Steven Reiss, Fall 2017 (Nick)
Upcoming Homework

- Read Lab handout
  - Install Arduino IDE
- Initial project proposal and budget
  - Make sure it’s something you are really interested in
  - It’s okay to be ambitious
    - It’s easier to scale down than to scale up
- Submit on Canvas by Monday 9/18 at 11:59pm
Arduino Programming

- C (Actually somewhat C++)
- Types
  - void, boolean, char, unsigned char, byte
  - int, unsigned int, word, short (16 bit)
  - long, unsigned long (32 bit)
  - float, double
Arduino Library Functions

- **I/O**
  - pinMode, digitalWrite, digitalRead
  - analogReference, analogRead, analogWrite

- **Time**
  - millis, micros, delay, delayMicroseconds

- **Standard C/C++ libraries (math.h, etc.)**
  - min, max, abs, constrain, map, pow, sqrt, sin, cos, tan

For more info: https://www.arduino.cc/en/Reference/HomePage
Arduino Programs

- **setup()**
  - Handle initialization
  - Setup I/O pins, clocks, interrupts

- **loop()**
  - Handle tasks
Arduino Program Structure

It’s really just a C program...

```c
#include <Arduino.h>
void main(void)
{
    initialize_system();
    setup();

    while(1) {
        loop();
    }
}
```

You provide these!
Arduino Program Example

```c
int led = 13;

void setup() {
    pinMode(led, OUTPUT);
}

void loop() {
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
    delay(1000); // wait for a second
}
```

- **introductory comments** describe the program
- **variable declaration section**
- **setup section**
- **loop section**
Clapper Code: setup()

```c
void setup()
{
    cur_state = STATE_OFF;                      // Initialize program state
    clap_done = 0;
    last_clap = 0;
    clap_count = 0;
    do_clap = 0;
    check_time = 0;
    pinMode(LIGHT0_PIN,OUTPUT);         // Configure pins
    pinMode(LIGHT1_PIN,OUTPUT);
    pinMode(SOUND_PIN,INPUT);
}
```
Clapper Code: loop()

```cpp
void loop()
{
    long time = millis();

    // Handle tasks
    handleLights(time);      // Update LEDs
    handleSound(time);       // Do clap detection
    handleClap();            // Change program state
}
```
Light Logic

- Lights are independent
  - Can be off, on, flashing on, flashing off

- Light structure
  - Tells us flash rate, output pin, state
  - Need to know when to change state
const int SOUND_PIN = 0;
const int LIGHT0_PIN = 13;
const int LIGHT1_PIN = 12;
const int THRESHOLD = 800;
const int FLASH0_TIME = 1000;
const int FLASH1_TIME = 500;
const int SETTLE_TIME = 10;
const int CLAP_TIME = 1000;
const int STATE_OFF = 0;
const int STATE_ON0 = 1;
const int STATE_ON1 = 2;
const int STATE_FLASH0 = 4;
const int STATE_FLASH1 = 8;

int clap_count = 0;
long clap_done = 0;
int last_clap = 0;
int cur_state = 0;
int orig_state = 0;
int do_clap = 0;
long flash0_time = 0;
long flash1_time = 0;
long check_time = 0;
Clapper Code: Light Definitions

typedef struct Light {
    int pin_number;
    int on_state;
    int flash_state;
    int flash_time;
    long update_time;
} Light;

struct Light  light_0 = {
    LIGHT0_PIN, STATE_ON0, STATE_FLASH0, FLASH0_TIME, 0 
};

struct Light  light_1 = {
    LIGHT1_PIN, STATE_ON1, STATE_FLASH1, FLASH1_TIME, 0 
};
Clapper Code: handleLights()

```c
void handleLights(long time)
{
    setLight(time,&light_0);
    setLight(time,&light_1);
}
```
Clapper Code: setLight()

```c
void setLight(long time, struct Light * l) {
    int l0 = LOW;
    if (((cur_state & l->on_state) != 0) l0 = HIGH;
    else if (((cur_state & l->flash_state) != 0) {
        if (l->update_time == 0) {
            l0 = HIGH;
            l->update_time = time + l->flash_time;
        } else if (l->update_time > 0) {
            if (time < l->update_time) l0 = HIGH;
            else {
                l->update_time = -(time + l->flash_time);
            }
        } else {
            if (time < -(l->update_time)) ;
            else {
                l->update_time = time + l->flash_time;
                l0 = HIGH;
            }
        }
    } else if (l->update_time > 0) {
        if (time < l->update_time) l0 = HIGH;
        else {
            l->update_time = -((time + l->flash_time));
            digitalWrite(l->pin_number, l0);
        }
    }
}
```
Clap Detection Logic Problems

- Clap may be longer than one cycle
- Input might be bouncy
- Need a gap between claps
  - Minimum and maximum
What Does a Clap Look Like

SHOW CLAP1 CODE
Clapper Code: Clap detection

```c
void handleSound(long time)
{
    if (check_time != 0 && time < check_time)
        return;
    check_time = 0;
    int val = analogRead(SOUND_PIN);
    if (val >= THRESHOLD) {
        if (last_clap == 0 && clap_count < 3) {
            if (clap_done == 0) orig_state = cur_state;
            else cur_state = orig_state;
            ++clap_count;
            clap_done = time + CLAP_TIME;
        }
        do_clap = 1;
        last_clap = 1;
    }
    check_time = time + SETTLE_TIME;
}
else last_clap = 0;
if (!do_clap && clap_done > 0 && time > clap_done) {
    clap_count = 0;
    clap_done = 0;
}
```
Clapper Code: Clapper logic

```c
void handleClap() {
    if (do_clap) {
        do_clap = 0;
        switch (clap_count) {
            case 0:
                break;
            case 1:
                if (cur_state == STATE_OFF) cur_state = STATE_ON0;
                else if (cur_state == STATE_ON0) cur_state = STATE_ON1;
                else cur_state = STATE_OFF;
                break;
            case 2:
                if (cur_state == STATE_OFF) cur_state = STATE_ON0 | STATE_ON1;
                else if (cur_state == STATE_ON0) cur_state = STATE_ON1;
                else if (cur_state == STATE_ON1) cur_state = STATE_ON0;
                else cur_state = STATE_OFF;
                break;
            case 3:
                if (cur_state == STATE_OFF) cur_state = STATE_FLASH0 | STATE_FLASH1;
                else if (cur_state == STATE_ON0) cur_state = STATE_FLASH0;
                else if (cur_state == STATE_ON1) cur_state = STATE_FLASH1;
                else cur_state = STATE_OFF;
                break;
        }
    }
}
```
Arduino Programming

Setup routine
- Called to initialize things
- Set direction of each pin
- Initialize state and other variables
- Set initial output values

9/25/17
Lecture 5: Arduino Programming
Arduino Programming

- **Loop routine**
  - Called continually by the OS
  - Does the various tasks
  - Handles frequency of tasks
    - Do some all the time
    - Do others every k times
    - Do others after elapsed time k
  - Should have a fixed max time
Arduino Environment

- Basic program = sketch
- Can be multiple files
- Save, check, download
Arduino Programming: Common pitfalls

- Handling multiple tasks can be tricky
  - Must avoid blocking, delay(), etc.
- No OS to help you
- Limited memory
  - Some C++ libraries available, but have high cost
  - Can easily overflow stack
Suggest a change to the clapper