Arduino Programming
Part 4: Flow Control

EAS 199B, Winter 2010

Gerald Recktenwald
Portland State University
gerry@me.pdx.edu

Goal

• Make choices based on conditions in the environment
• Logical expressions: Formulas that are T or F
• Different kinds of choices
  ❖ Act on a single condition
  ❖ Choose one course of action from several

Conventional on/off switch

Basic light switch or rocker switch
  ❖ Makes or breaks connection to power
  ❖ Switch stays in position: On or Off
  ❖ Toggle position indicates the state
  ❖ NOT in the Arduino Inventors Kit
Momentary or push-button switches

- Temporary “click” input
- Normally open
  - electrical contact is made when button is pressed
- Normally closed
  - electrical contact is broken when button is pressed
- Internal spring returns button to its un-pressed state

Momentary Button and LED Circuit

Digital input with a pull-down resistor
- When the switch is open:
  - Digital input pin is tied to ground
  - No current flows, so there is no voltage difference from input pin to ground
  - Reading on digital input is LOW
- When the switch is closed:
  - Current flows from 5V to ground, causing LED to light up.
  - The 330Ω resistor limits the current draw by the input pin.
  - The 10k resistor causes a large voltage drop between 5V and ground, which causes the digital input pin to be closer to 5V
  - Reading on digital input is HIGH

Programs for the LED/Button Circuit

1. Continuous monitor of button state
   - Program is completely occupied by monitoring the button
   - Used as a demonstration — not practically useful
2. Wait for button input
   - Blocks execution while waiting
   - May be useful as a start button
3. Interrupt Handler
   - Most versatile
   - Does not block execution
   - Interrupt is used to change a flag that indicates state

All three programs use the same circuit
Continuous monitor of button state

This program does not control the LED

```c
int button_pin = 4;  // pin used to read the button

void setup() {
  pinMode(button_pin, INPUT);
  Serial.begin(9600);  // Button state is sent to host
}

void loop() {
  int button;
  button = digitalRead(button_pin);
  if (button == HIGH) {
    Serial.println("on");
  } else {
    Serial.println("off");
  }
}
```

Serial monitor shows a continuous stream of “on” or “off”

Basic “if” construct

- Condition that is TRUE or FALSE
- Brackets define limits of code block
- Code block

Example:

```c
button_value = digitalRead(button_pin);
if (button_value == HIGH) {
  digitalWrite(LED_pin, HIGH);
}
```
Comparison operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Is less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Is greater than than</td>
</tr>
<tr>
<td>==</td>
<td>Is equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Is greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Is less than or equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Is not equal to</td>
</tr>
</tbody>
</table>

Practice

What is the value \(z\)?

```c
x = 2;
y = 5;
if ( x < y ) {
    z = y - x;
}
```

```c
x = 2;
y = 5;
if ( x > y ) {
    z = y - x;
}
```

Basic “while” construct

Condition that is TRUE or FALSE

Brackets define limits of code block

keep doing something while condition is true

Code block

Example:

```c
int x;
x = analogRead(2);
while ( x > 355 ) {
    digitalWrite( LED_pin, OFF);
    x = analogRead(2);
}
```
### Wait for button input

```c
int button_pin = 4;       // pin used to read the button

void setup() {
  int start_click = LOW;       // Initial state: no click yet
  pinMode( button_pin, INPUT);
  Serial.begin(9600);
  while ( !start_click ) {
    start_click = digitalRead( button_pin );
    Serial.println("Waiting for button press");
  }
}

void loop() {
  int button;
  button = digitalRead( button_pin );
  if ( button == HIGH ) {
    Serial.println("on");
  } else {
    Serial.println("off");
  }
}
```

### Other references

**Ladyada tutorial**
- Excellent and detailed
- [http://www.ladyada.net/learn/arduino/lesson5.html](http://www.ladyada.net/learn/arduino/lesson5.html)

**Arduino reference**
- Minimal explanation
- Using interrupts