CS 5422: Physical Computing

State-based Design

Slides will be available through Blackboard. There is no textbook.

Standard Arduino Loop

A natural way to handle I/O in an Arduino system:

```
while (1) {
  if (input0_ready) {
    operation0;
  } else if (input1_ready) {
    operation1;
  } ...
  if (produce_output0) {
    output0;
  } ...
}
```

Standard Arduino Loop

Efficiency of the system?

- Response time
- Power consumption

Battery ratings:

- Sample cell phone: 1400 mAh
- Amazon Kindle: 5.7 Wh

State-based Embedded Software

Another way to think of the system is to view it as follows:

```
while (1) {
  wait for any interrupt;
  if (interrupt0) {
    operation0;
  } else if (interrupt1) {
    operation1;
  } ...
}
```

State-based Embedded Software

How do we implement this?

- Setup interrupt handlers for each interrupt type
- Interrupt handlers execute the specified operation

What happens if an operation is very long?

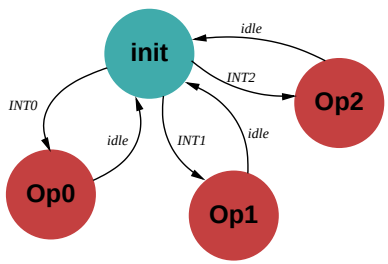
State-based Embedded Software

Some operations should be interruptible, while others are not.

- Create a *state machine*
- In each state, determine which interrupts must be handled/ignored
- Interrupts or completion of handlers correspond to state transitions

State-based Embedded Software

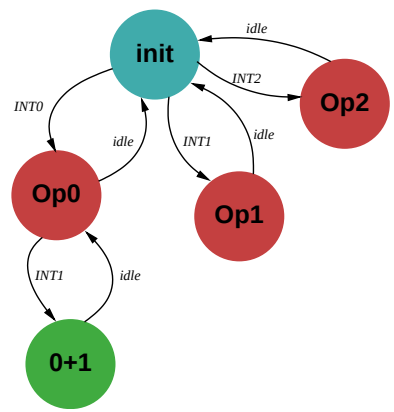
Example:



How do you translate this into a program?

State-based Embedded Software

Example:



How do you translate this into a program?

Sleep Modes

Standard scenario in embedded systems:

- Waiting processes only
- Processes are waiting for an external interrupt

Example: Phones, thermostat control, . . .

Running a loop is a waste of power.
(Sometimes called the “idle loop.”)

Sleep Modes

Most microcontrollers have “sleep modes”

- Special hardware support
- Minimize power consumption
- Stop running the processor
- Resume on an interrupt

Often there are multiple sleep modes (depending on how much of the system is inactive).