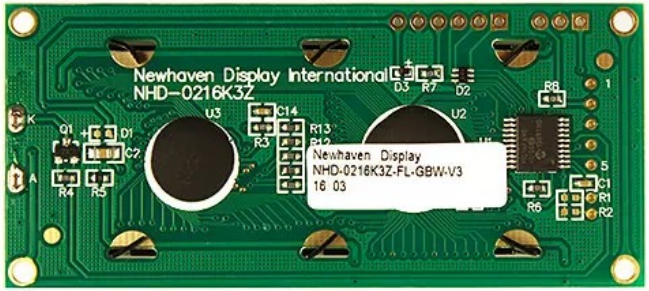
PAT 451 INTERACTIVE MEDIA DESIGN I

ARDUINO_STATES_LCD

LCD



- Liquid Crystal Display
- Ours is made by Newhaven Display, model

  NHD-0216K3Z-FL-GBW

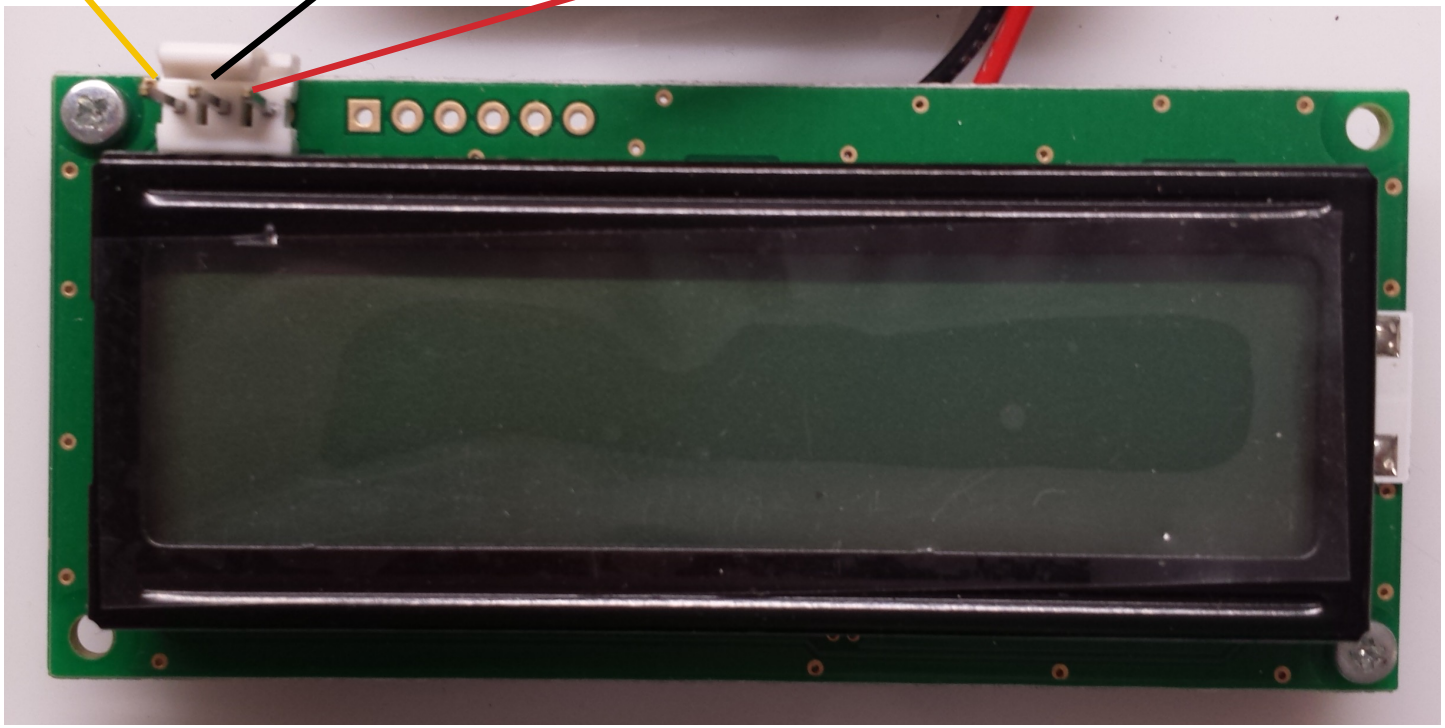
- Interface to the display is Serial
  - this is unusual, and desirable because it uses less I/O pins
  - We will talk about protocols in a future class

Connecting the LCD

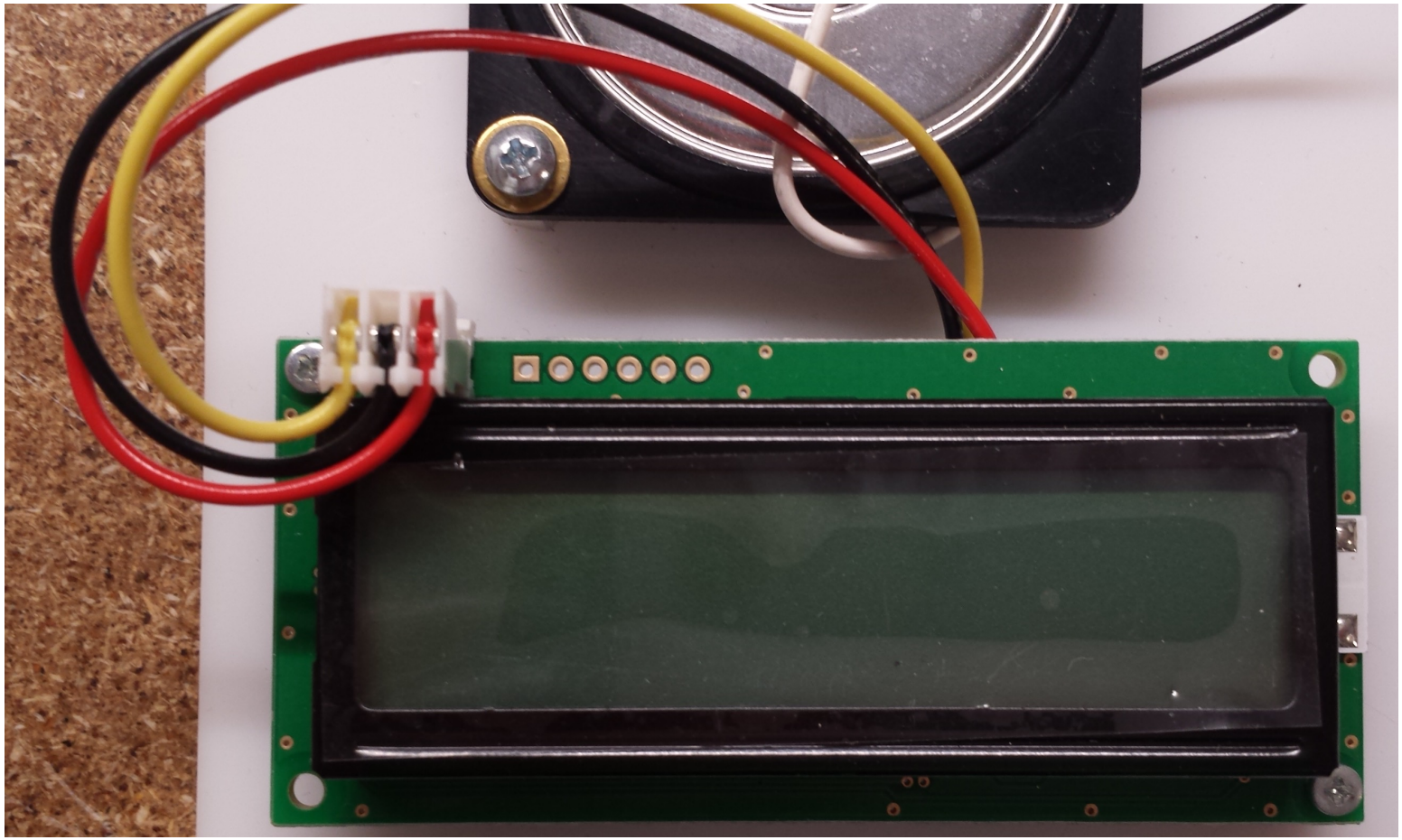
Pin 1: Rx (Serial Receive)
Yellow wire

Pin 2: VSS (Ground)
Black wire

Pin 3: VDD (Power Supply +5V)
Red wire



Connecting the LCD



USING THE LCD

1. Copy the NewhavenLCDSerial directory to:
   ~/Documents/Arduino/libraries
2. Connect the power and ground pins
3. Connect the Serial data line to a digital pin

CODE

```
#include <NewhavenLCDSerial.h>
// at top of program. Includes the library to allow you to
access the LCD functions.

NewhavenLCDSerial lcd(pin);
// Initialize LCD using pin pin
// creates an object of class NewhavenLCDSerial, named lcd
```

USING THE LCD

```
// Methods of class NewhavenLCDSerial
// if your object is called lcd, you would use these like
// lcd.clear();

void clear();         // clear the screen
void home();          // move cursor to home position (0,0)
void setCursor(int row, int col);
// move cursor to position row, col

void setBrightness(int);   // backlight brightness (1-8)
void setContrast(int);     // display contrast (1-50)

void display();       // display on
void noDisplay();     // display off
```

PRINTING TO THE LCD

```
void print("string"); // print a string literal
void print('c');      // print a character literal
void print(12);       // print the number 12
void print(int);      // print an integer variable
void print(int,BIN);  // print an integer in binary
void print(float);    // print a floating point number
                     // (defaults to 2 decimal places)

void print(float,n);  // print a floating point number
                      // to n decimal places
```

Print methods are the same as those available in the Serial class. See:
https://www.arduino.cc/en/Serial/Print

LCD CURSOR COMMANDS

```
void blink();          // blinking cursor
void noBlink();        // unblinking cursor
void cursor();         // underline cursor
void noCursor();       // no cursor
void moveCursorLeft();
void moveCursorRight();
void backspace();

void moveLeft();       // move display left
void moveRight();      // move display right
```

"Multitasking" Approach to Arduino

Feijs, L. 2013. "Multi-tasking and Arduino: Why and How?" in *Proceedings of the Conference on Design and Semantics of Form and Movement*, pp. 119-127.

Main idea:

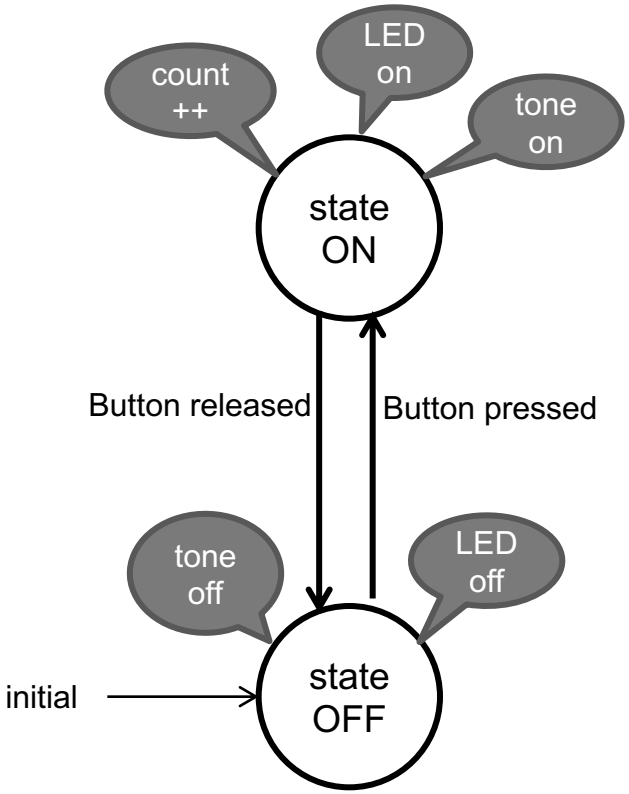
- **Create a function for each "task" the Arduino performs**
- **Call these functions in rapid succession in main loop**
- **Use State Transition Diagrams**

RECALL:

Write a program that:

- Detects a button
- Lights an LED for as long as the button is held down
- Plays a tone for as long as the button is held down
- Uses `Serial.println()` to send a number each time the button is pressed. It should start at 0 and count up, only sending a single new value at the onset of each press.

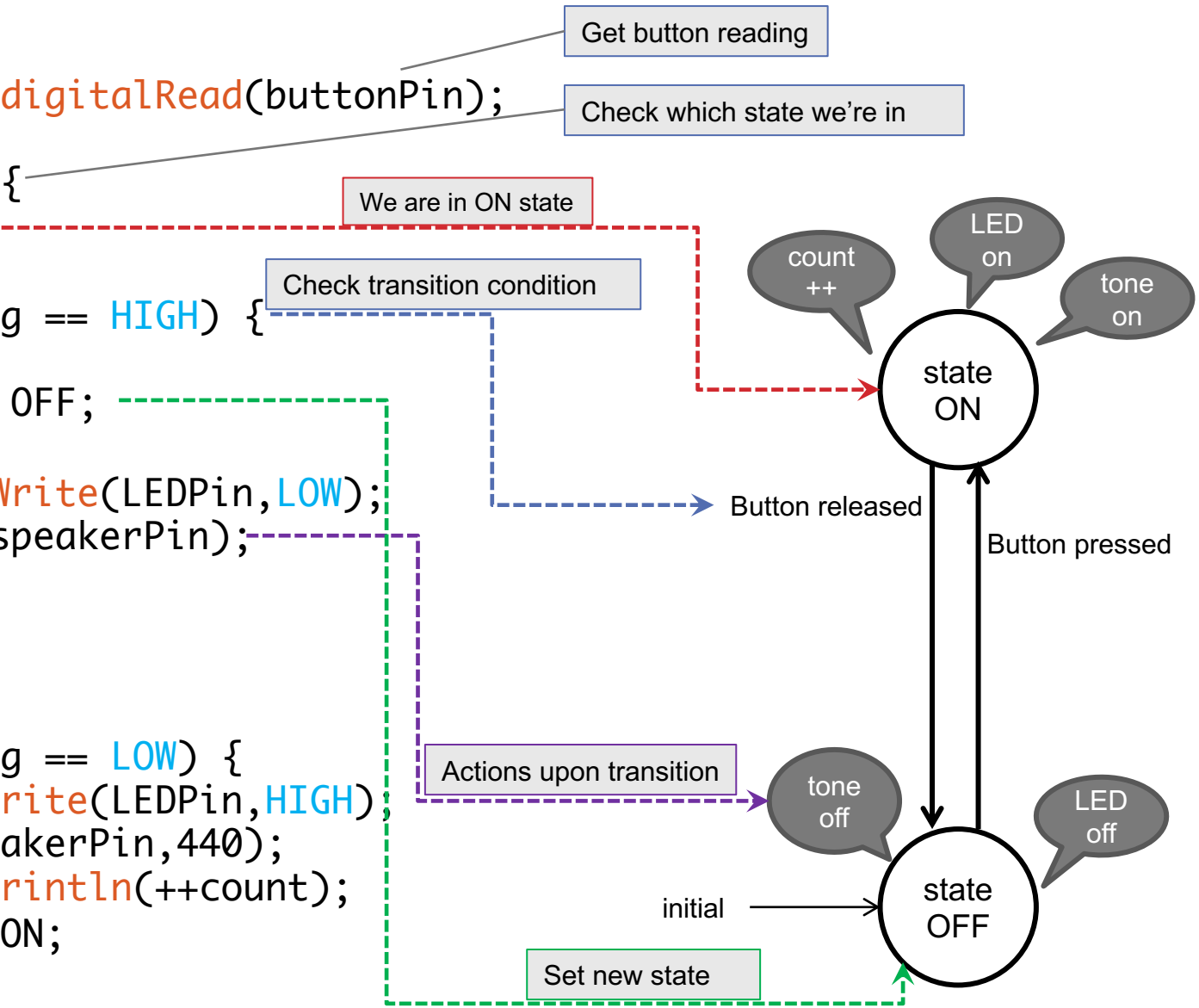
STATE TRANSITION DIAGRAM



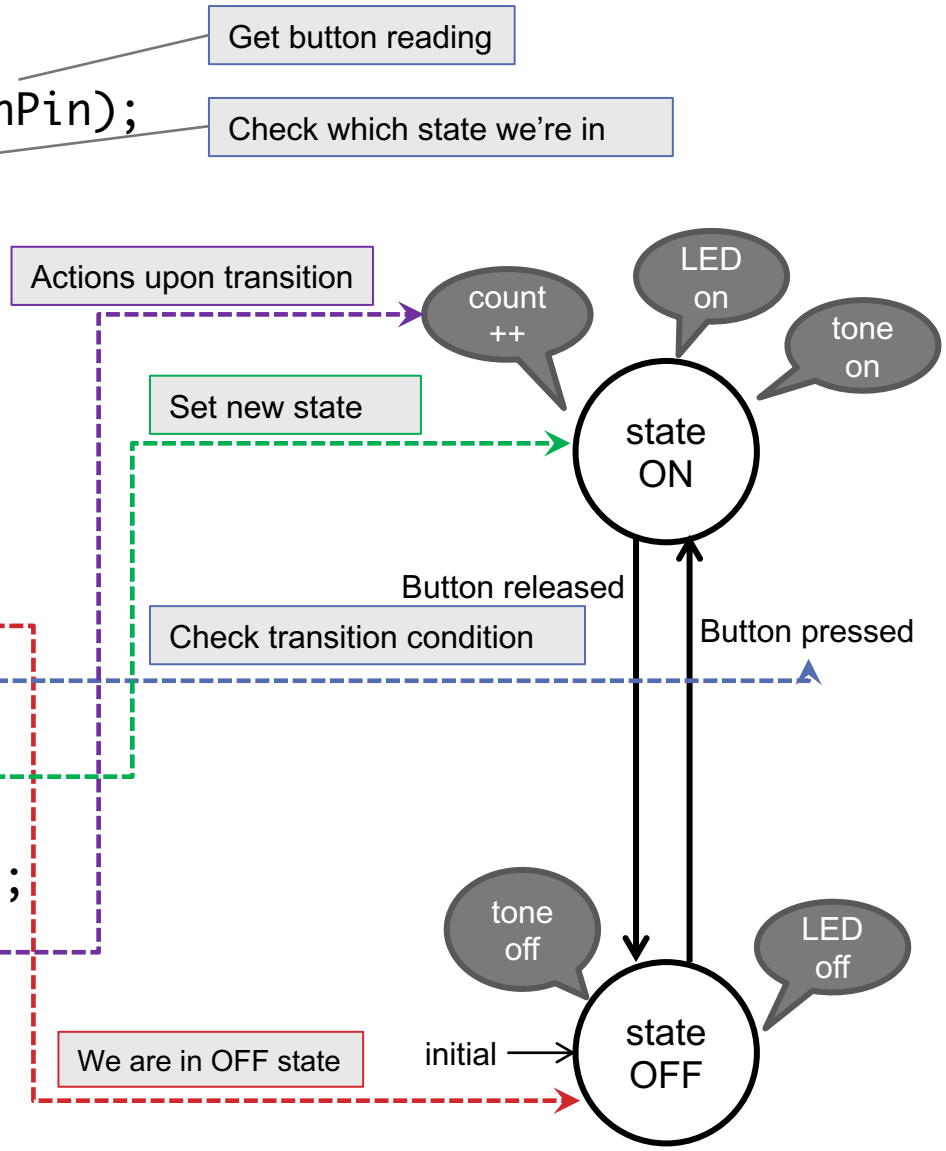
STRUCTURE OF A SWITCH STATEMENT

```
switch (var) {
  case 1:
    //do something when var equals 1
    break;
  case 2:
    //do something when var equals 2
    break;
  default:
    // if nothing else matches, do the default
    // default is optional
    break;
}
```

```
void loop() {

  int reading = digitalRead(buttonPin);

  switch(state) {
    case ON:

      if (reading == HIGH) {

        state = OFF;

        digitalWrite(LEDPin,LOW);
        noTone(speakerPin);

       } break;

    case OFF:
      if (reading == LOW) {
        digitalWrite(LEDPin,HIGH);
        tone(speakerPin,440);
        Serial.println(++count);
        state = ON;
      } break;

  }
}
```



```
void loop() {

  int reading = digitalRead(buttonPin);

  switch(state) {
    case ON:
      if (reading == HIGH) {
        state = OFF;
        digitalWrite(LEDPin,LOW);
        noTone(speakerPin);
       } break;

    case OFF:
      if (reading == LOW) {

        state = ON;

        digitalWrite(LEDPin,HIGH);
        tone(speakerPin,440);
        Serial.println(++count);
       } break;

  }
}
```



USING ENUM FOR STATE VARIABLES

```
enum cardsuit { HEARTS, DIAMONDS, CLUBS, SPADES };
```

enum defines a new type

Arbitrary name for our new type

List of possible values the type can take

```
cardsuit mycard = CLUBS;
```

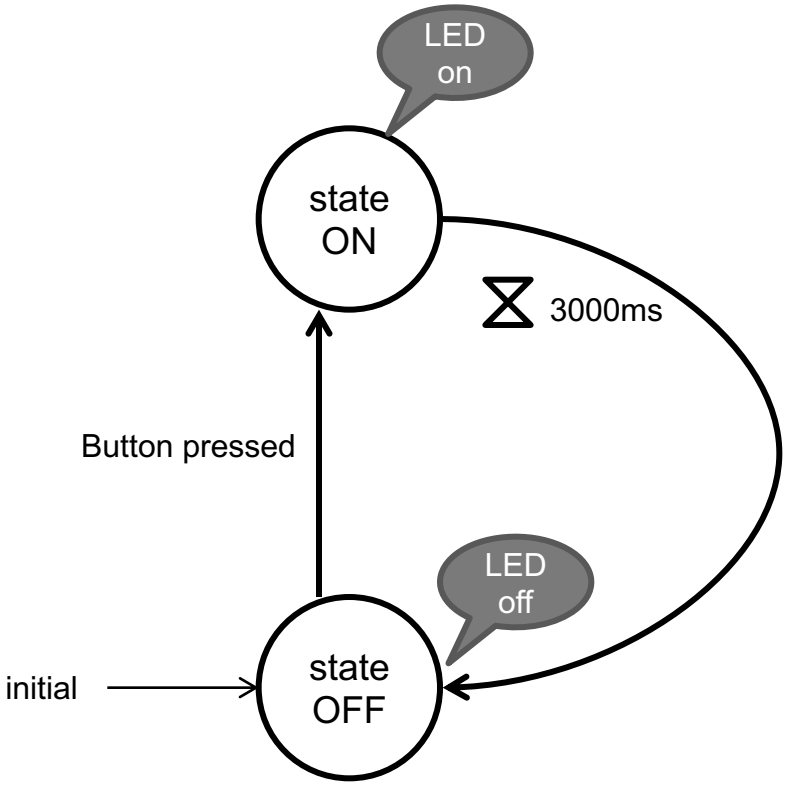
Create a new variable of type cardsuit and assign its initial value to be CLUBS.

EXERCISE 1:

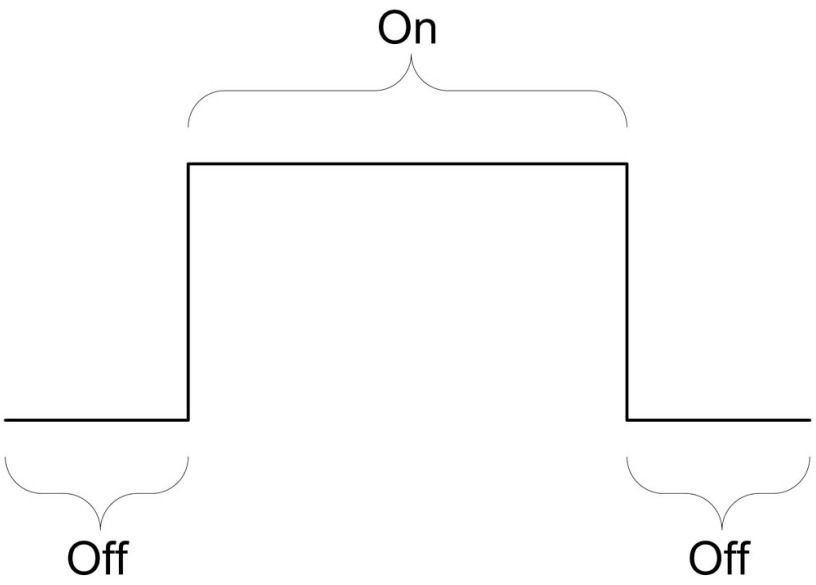
Create a state-transition diagram for the following system:

- 1 button, 1 LED, 1 speaker
- LED and speaker are initially off
- When user presses button, LED turns on, speaker plays a tone
- After 3 seconds of initial press, LED turns off, speaker stops

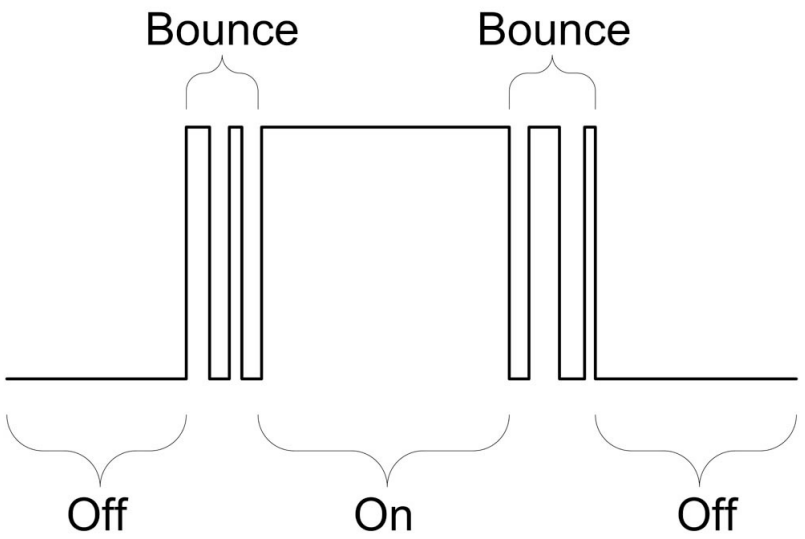
TIMED LIGHT SWITCH



BUTTON BOUNCING



IDEAL



REALITY

DEBOUNCING EXERCISE

Create a state transition diagram for a program that:

- Detects a button press --- with debouncing!

- Lights an LED for as long as the button is held down
- Plays a tone for as long as the button is held down
- Increments a counter each time the button is pressed

DEBOUNCING WITH STATES