```
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#include <FluxGarage_RoboEyes.h>
roboEyes roboEyes; // create RoboEyes instance
// Button pin definitions
#define BUTTON_HAPPY 2
#define BUTTON_ANGRY 3
#define BUTTON_TIRED 4
#define BUTTON_CONFUSED 5

String currentEmotion = "DEFAULT";
void setup() {
 Serial.begin(9600);

 // Startup OLED Display
 // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
 if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C or 0x3D
  Serial.println(F("SSD1306 allocation failed"));
  for(;;); // Don't proceed, loop forever
 }
```

```
  // Startup robo eyes
  roboEyes.begin(SCREEN_WIDTH, SCREEN_HEIGHT, 100); // screen-width, screen-height, max framerate

  // Define some automated eyes behaviour
  roboEyes.setAutoblinker(ON, 3, 2); // Start auto blinker animation cycle -> bool active, int interval, int variation -> turn on/off, set interval between each blink in full seconds, set range for random interval variation in full seconds
  roboEyes.setIdleMode(ON, 2, 2); // Start idle animation cycle (eyes looking in random directions) -> turn on/off, set interval between each eye repositioning in full seconds, set range for random time interval variation in full seconds

  // Define eye shapes, all values in pixels
  roboEyes.setWidth(36, 36); // byte leftEye, byte rightEye
  roboEyes.setHeight(36, 36); // byte leftEye, byte rightEye
  roboEyes.setBorderradius(8, 8); // byte leftEye, byte rightEye
  roboEyes.setSpacebetween(10); // int space -> can also be negative

  // Cyclops mode
  //roboEyes.setCyclops(ON); // bool on/off -> if turned on, robot has only on eye

  // Define mood, curiosity and position
  roboEyes.setMood(DEFAULT); // mood expressions, can be TIRED, ANGRY, HAPPY, DEFAULT
  roboEyes.setPosition(DEFAULT); // cardinal directions, can be N, NE, E, SE, S, SW, W, NW, DEFAULT (default = horizontally and vertically centered)
  roboEyes.setCuriosity(ON); // bool on/off -> when turned on, height of the outer eyes increases when moving to the very left or very right

  // Set horizontal or vertical flickering
  roboEyes.setHFlicker(ON, 2); // bool on/off, byte amplitude -> horizontal flicker: alternately displacing the eyes in the defined amplitude in pixels
  roboEyes.setVFlicker(ON, 2); // bool on/off, byte amplitude -> vertical flicker: alternately displacing the eyes in the defined amplitude in pixels

  // Play prebuilt oneshot animations
  roboEyes.anim_confused(); // confused - eyes shaking left and right
  roboEyes.anim_laugh(); // laughing - eyes shaking up and down



  // Configure buttons as input with pull-up resistors
  pinMode(BUTTON_HAPPY, INPUT_PULLUP);
  pinMode(BUTTON_ANGRY, INPUT_PULLUP);
  pinMode(BUTTON_TIRED, INPUT_PULLUP);
  pinMode(BUTTON_CONFUSED, INPUT_PULLUP);

  Serial.println("System ready! Press buttons to change emotions.");
} // end of setup



void loop() {
  // Check button states
  if (digitalRead(BUTTON_HAPPY) == LOW) { // Button is pressed
```

```
    currentEmotion = "HAPPY";
    Serial.println("Happy mode activated!");
    roboEyes.setMood(HAPPY);
  }
  else if (digitalRead(BUTTON_ANGRY) == LOW) {
    currentEmotion = "ANGRY";
    Serial.println("Angry mode activated!");
    roboEyes.setMood(ANGRY);
  }
  else if (digitalRead(BUTTON_TIRED) == LOW) {
    currentEmotion = "TIRED";
    Serial.println("Tired mode activated!");
    roboEyes.setMood(TIRED);
  }
  else if (digitalRead(BUTTON_CONFUSED) == LOW) {
    currentEmotion = "CONFUSED";
    Serial.println("Confused animation activated!");
    roboEyes.anim_confused(); // Play the confused animation
  }
 roboEyes.update(); // update eyes drawings
 // Dont' use delay() here in order to ensure fluid eyes animations.
 // Check the AnimationSequences example for common practices.
}
```