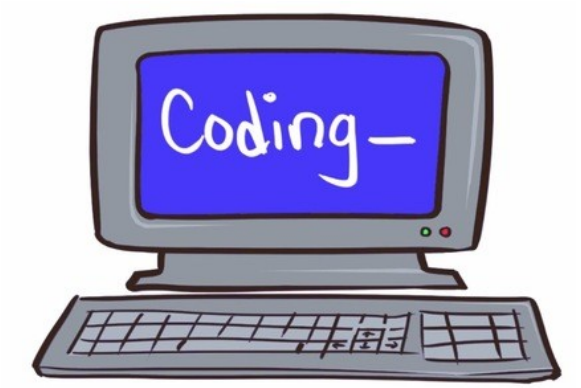


Class 46

Classes & Objects

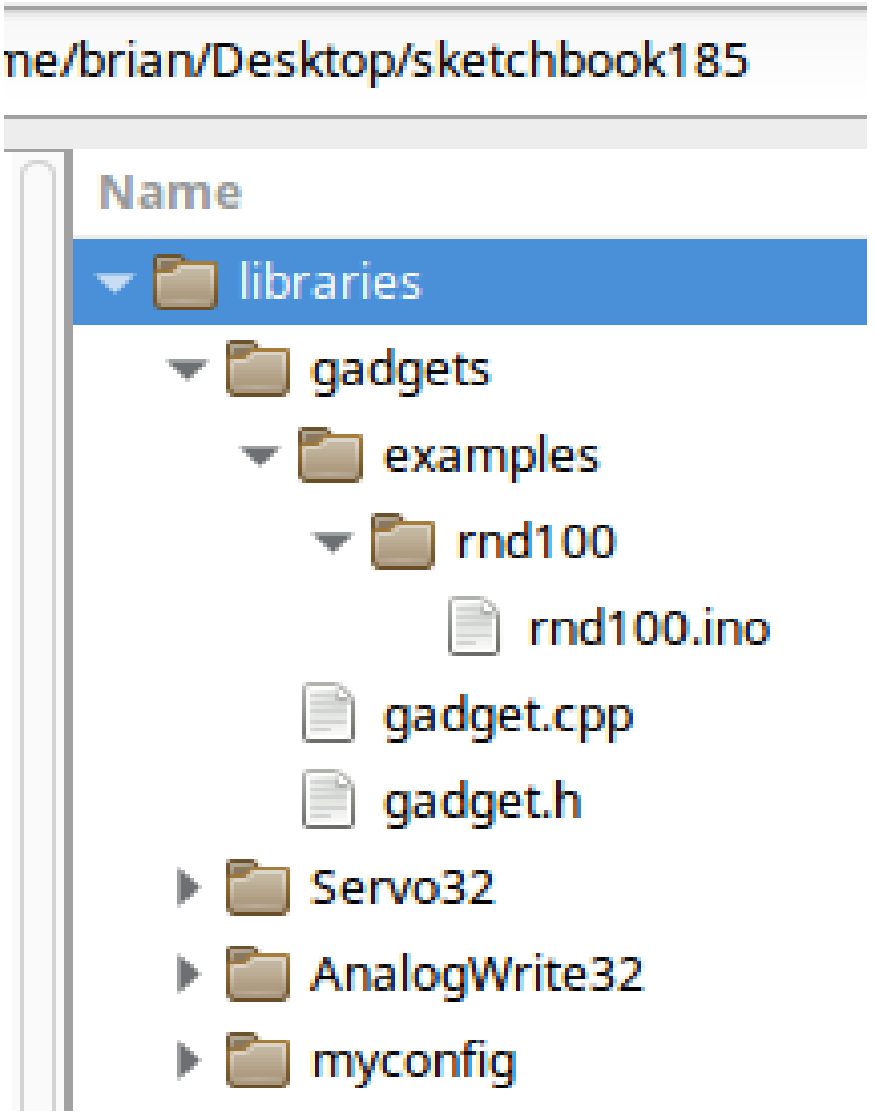
...and finally
Integrating ALL our devices into one mega-sketch

The ROBOT:
Hardware we have been assembling:

- RGB LED
- “ERC” generic gpio controller
- Servomotors
- Oled screen
- Ultrasonic Sonar
- (Easy-) Wifi
- Blynk control from phone
- H-bridge motor driver
- Wheel motor battery supplies
- Infrared “line-follower”
- Pitch / Roll / Compass (IMU)
- IR Remotes
- Wheel encoders

Simplest class library

gadgets library folder
gadget.h/cpp
rnd100.ino



Our servo again

servo2.ino | AnalogWrite32.h | AnalogWrite32.cpp | servo32.h | servo32.cpp

```
#include "myconfig.h"
#include <AnalogWrite32.h>

#define SERVO_CALIB 11

void setup()
{
  Serial.begin(SERIAL_BAUD);
  analogWrite15(SERVO1, 0, 50);
  analogWrite15(SERVO2, 0, 50);
}

void writeServo(int pin, int degr) {
    analogWrite15 (pin, ((degr-90)*SERVO_CALIB)+2500) ;
}

void loop()
{
  for (int i = 1; i<10; i++) {
      writeServo(SERVO1, 0);
      writeServo(SERVO2, 90);
      Serial.println("0  90");
      delay(1000);
      writeServo(SERVO1, 180);
      writeServo(SERVO2, 0);
      Serial.println("180  0");
      delay(1000);
  }
  analogWrite15(SERVO1, 0);  // == stop, zero dutycycle
  analogWrite15(SERVO2, 0);
  for (;;){}
}
```

Bring in library with new 15bit analog

A scaling to stretch the “span”

Attach & Init analog service at no dutycycly

We “scale” the degrees to the corresponding duty cycle

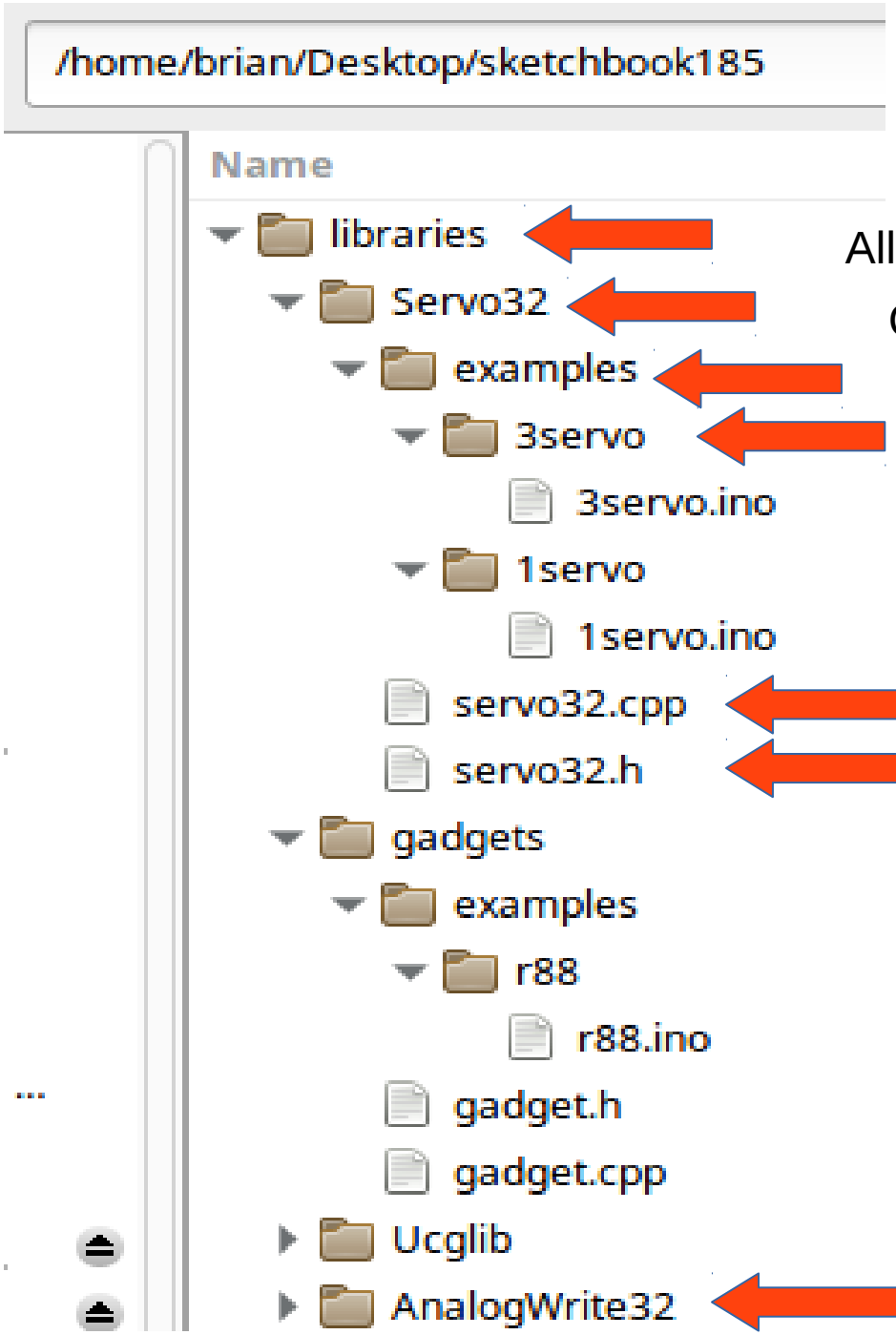
Write a degree setting to servo

Servo2.ino

Classes & Objects

Let's convert this simple sketch servo2.ino
Into “object oriented” form.

A Servo class & Servo object



All libraries go in “libraries” folder

Our new servo lib for esp32

Optional example sketches for this library

Code cpp file(s) of this library

The main “header” file of this library

Our servo32 lib needs this lib too

```
#include <servo32.h>
#include "myconfig.h"

Servo svo;

void setup()
{
    Serial.begin(SERIAL_BAUD);
    svo.attach(SERVO1);
}

void loop()
{
  for (int i = 1; i<10; i++) {
      svo.write(0);
      Serial.println("0");
      delay(1000);
      svo.write(180);
      Serial.println("180");
      delay(1000);
  }
  svo.stop();
  Serial.println("Stopped");
  for (;;)
      {}
}
```

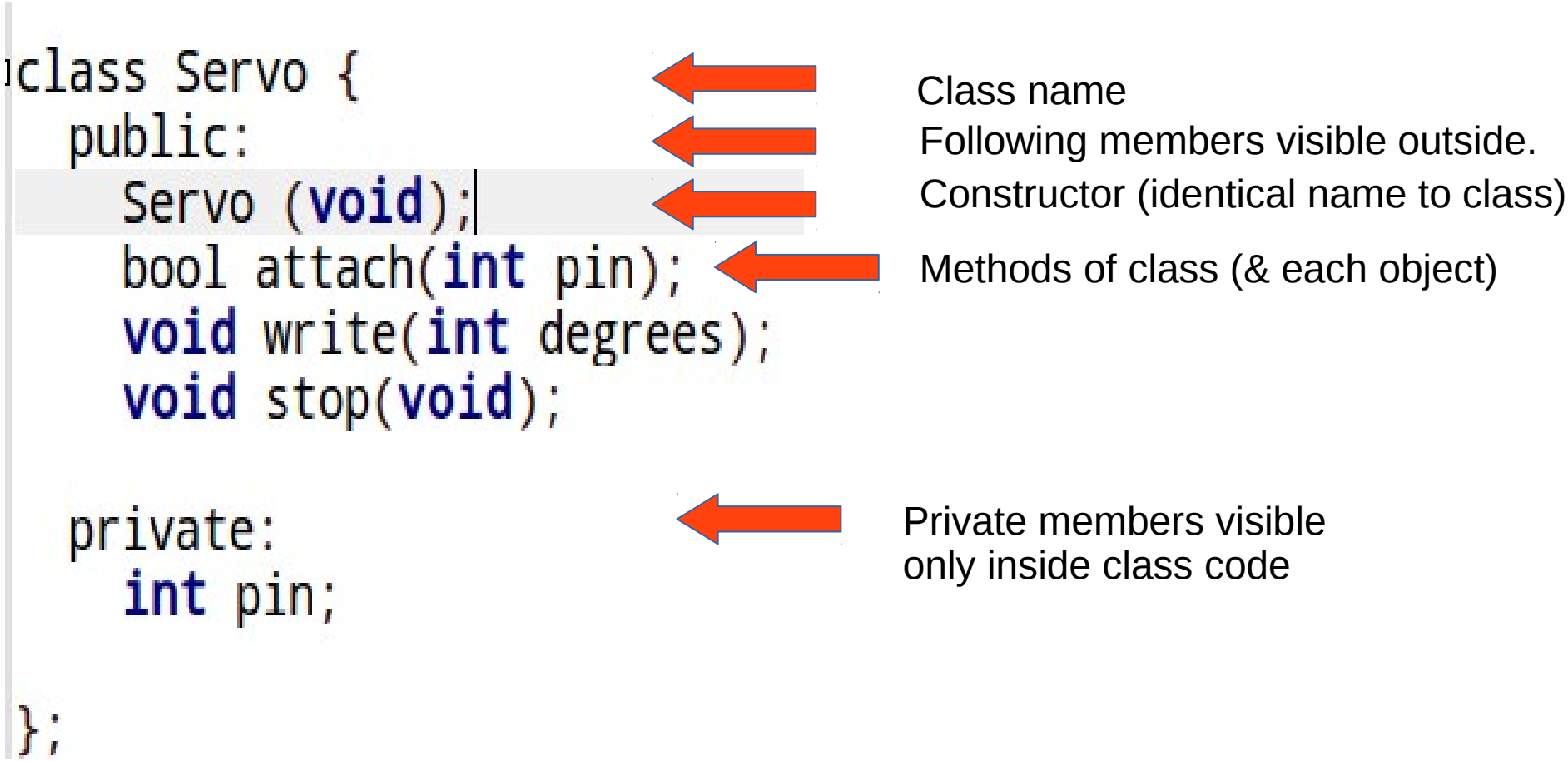
Include the lib header file. That grabs all the library incl its cpp files.

"Constructor". Create one object ("svo") from Servo class

Do any setup in object svo.

Do any activity using functions (methods) of svo

Servo32.h Library header file



Every derived object (“instance”) gets its own methods/properties
As defined above for the class.

Servo32.cpp

```
#include <servo32.h>

bool Servo::attach(int pin){

};

void Servo::write(int degrees){

};

void Servo::stop(void){

};
```

Include own header

```
#include <servo32.h>
#include <AnalogWrite32.h>

Servo::Servo (void) {
    // nothing to do.  But we still shoud put a "constructor" here!
};

bool Servo::attach(int gpiopin, int calib){
    pin = gpiopin;
    servo_calib = calib;
    analogWrite15(pin, 0, 50);
};

void Servo::write(int degrees){
    analogWrite15(pin, ((degrees-90)*servo_calib)+2500);
};

void Servo::stop(void){
    analogWrite15(pin, 32768);

};
```

Robot0.ino

Our first sketch attempting to roll together
ALL the devices we have been developing

```
#include "myconfig.h"
#include "easywifi.h"
#include "SSD1306.h"
#include "easyoled.h"
#define BLYNK_PRINT Serial
#include <BlynkSimpleEsp32.h>
#include <L298N.h>
#include <Servo.h>
#include <IRremote.h>
#include <QuadEncoder.h>
#include "HCSR04.h"
#include "easyIMU.h"
#include "ZTimer.h"

SSD1306  display(OLEDADDR, SDA, SCL);

//create 2 motor instances
L298N wheelL(Aen, Ain1, Ain2);
L298N wheelR(Ben, Bin1, Bin2);

// Create 2 servo objects called servo1, servo2
Servo servo1;
Servo servo2;

IRrecv irrecv(IRRX, LED_BUILTIN);
decode_results results;

ZTimer Timer500, Timer2000;
QuadEncoder counter1, counter2;

HCSR04 sonars[2];
SonarRotator sonarrotator(2, sonars);

OLED oled(&display);


void setup()
{
```

Robot0.ino

Integrating:

- #include library
- Object constructors
- Object initialising (setup)
- Any .run() functions (loop)
- Read/control each device (loop)