

A teal background with a repeating pattern of faint, white line-art icons. The icons include various tools and objects: a wrench, a screwdriver, a paintbrush, a hammer, a saw, a drill, a pencil, a ruler, a compass, a pair of pliers, a tape measure, a level, a square, a circle, a triangle, a rectangle, a hexagon, an octagon, a star, a heart, a flower, a leaf, a bird, a fish, a cat, a dog, a person, a car, a house, a tree, a mountain, a sun, a moon, a cloud, a rain cloud, a lightning bolt, a fire, a water drop, a leaf, a flower, a bird, a fish, a cat, a dog, a person, a car, a house, a tree, a mountain, a sun, a moon, a cloud, a rain cloud, a lightning bolt, a fire, a water drop.



code

SLQ Wiki Fabrication Lab 2024/04/19 05:19

```
// Version 0.14
// Now with servo and new neopixel library
// Two 'weapon' support.
// More code tidyups
// Robot jumper mode select

// USB Host shield
#include <PS3BT.h>
#include <usbhub.h>
#include <SPI.h>

// Built in servo library
#include <Servo.h>

// Uncomment one of these
// #include <Adafruit_NeoPixel.h>
#include <WS2812.h> // https://github.com/cpldcpu/light_ws2812

// Uncomment depending on style of bot
// #define SKIDBOT
// #define STEERBOT
// 0=Test only. 1=Skidsteer bot, 2=Steering style bot
char bottype=1;

// Setup serial port and debug timer
#define SERIAL_PORT_SPEED 115200
#define DEBUGVALUE 500 // Every DEBUGVALUE trigger debug code
unsigned long debugtimer;
unsigned long loopmills;

// Reduces controller jitter
#define DEADZONE 8

// H-Bridge motor controller for DC motors
// motor one
#define pwmA 5 // Must be a PWM pin
#define aIN1 2 // H-Bridge
#define aIN2 3 // Control pins
// motor two
#define pwmB 6 // PWM
#define bIN1 4
#define bIN2 7

// Servos
#define SERV00NEPIN 8 // Where are they connected
#define SERV0TWOPIN A4
Servo servo_one; // create servo object to control a servo
Servo servo_two;
```

```
// Battery monitoring analog input pin
#define VINPIN A0

int dirval,speedval,weppval,auxval;           // Inputs
int left_motor,right_motor,webb_motor,aux_motor; // Outputs

int motcurrmax;           // Peek control value
int battvolt;            // Battery voltage

// Spinner style weapon spinup delays. Prevents motor stripping and looks
// more dramatic
#define WEPPTIMERVALUE 100 // How often (in ms) to update speed
#define WEPPRATEDELTA 10   // Weapon spin up speed, units per WEPPTIMERVALUE
#define WEPPMAX 200        // Weapon max speed
unsigned long wepptimer;
int currweppspeed;

// Settings and timeouts for 'boost'.
// Motors are 6v and this allows tempory driving to full battery voltage
#define HALFPOWERTIMEOUT 10000 // In milis.
#define HALFPOWERCOOLDOWN -20000
#define HALFPOWER 170 // (actually halfish) Limit motor to 2/3 power:
(255/3)*2
long halfpowertimer;
int motormax=HALFPOWER;

// 0=Debug: No movement, all controllls displayed.
// 1=Single stick: Right stick, X axis Steer. Y axis throttle.
// 2=Two stick: Left stick Y axis Throttle. Right stick X axis steering.
// 3=D-Pad: For crazy people. Digital only for now, May add preassure sense.
// (Not working)
// 4=Hayden: Right trigger forward, left trigger backward, left stick X
// steering.
int controlmode=1;

// Main RGB LEDs
#define PIXELPIN A1
#define NUMPIXELS 2
//Adafruit_NeoPixel neopixel = Adafruit_NeoPixel(NUMPIXELS, PIXELPIN,
//NEO_GRB + NEO_KHZ400);
WS2812 LED(NUMPIXELS); // light_ws2812

// USB Host Shield object setup
USB Usb;
BTD Btd(&Usb); // You have to create the Bluetooth Dongle instance like so
PS3BT PS3(&Btd); // This will just create the instanceinstance

// Runs just once
```

```
void setup() {  
  
    // Test to see if Tx and Rx have been shorted by a jumper.  
    // This sets the robot's mode: Jumper off: Skid. Jumper on: Steer  
    pinMode(1, OUTPUT);          // Tx pin  
    pinMode(0, INPUT_PULLUP);    // Rx pin  
    setPixelRGB(0, 00, 30, 00);  
    digitalWrite(1, LOW);        // Pull Tx LOW  
    delay(10);  
    if(digitalRead(0)==LOW) {    // Read Rx  
        digitalWrite(1, HIGH);  // If low, take Tx HIGH  
        delay(10);  
        if(digitalRead(0)==HIGH) { // See if state changed  
            bottype=2;          // Change robot mode.  
        }  
    }  
    // Serial port setup with delay to connect  
    Serial.begin(SERIAL_PORT_SPEED);  
    delay(500);  
    Serial.print("Starting... Bot Mode: "); // Just so we know where we are  
    Serial.println(bottype,DEC);  
    // RGB led library setup  
    // neopixel.begin();  
    LED.setOutput(PINLEDPIN);  
  
    // Lights depend on bot type to show we have started  
    switch (bottype){  
        case 0:    // Test/debug  
            setPixelRGB(0, 30, 30, 30); // White  
            break;  
        case 1:    // Skidbot  
            setPixelRGB(0, 30, 30, 00); // Green  
            break;  
        case 2:    // Steerbot  
            setPixelRGB(0, 00, 30, 30); // Yellow  
            break;  
    }  
    setPixelRGB(1, 00, 00, 00);  
  
    // Give the user a moment to see the LEDs  
    delay(1000);  
  
    // set all the motor control pins to outputs and make sure motors are off  
    pinMode(pwmA, OUTPUT);  
    pinMode(pwmB, OUTPUT);  
    pinMode(aIN1, OUTPUT);  
    pinMode(aIN2, OUTPUT);  
}
```

```
pinMode(bIN1, OUTPUT);
pinMode(bIN2, OUTPUT);
setmotor(0,0);
setmotor(1,0);

// attach each servo to the servo object and set servos to halfway
servo_one.attach(SERV00NEPIN);
servo_two.attach(SERV0TWOPIN);
setservo(0,0);
setservo(1,0);

// Start USB subsystem
if (Usb.Init() == -1) {
    Serial.print("\r\nOSC did not start");    // Very bad. You will not robot
today
    setPixelRGB(0, 255, 00, 00);    // both lights red
    setPixelRGB(1, 255, 00, 00);
    while (1);    // halt - no point going further, hardware fault
}
Serial.print("\r\nWaiting for pairing..");    // Actually, just 'usb is
working'
}

// Around and around and around
void loop() {
    // For debug and weapon timeouts
    loopmills=millis();

    // Uncomment only to test motors. This disables everthing else
    // dcmot_test();

    // Make sure this runs fairly often. USB and bluetooth both need this
    Usb.Task();
    // No bluetooth connection
    if (!PS3.PS3Connected) {
        //turn off or center all motors
        setmotor(0,0);
        setmotor(1,0);
        setservo(0,0);
        setservo(1,0);
        // Turn on blue LED, others off
        setPixelRGB(0, 00, 00, 50);
        setPixelRGB(1, 00, 00, 00);
        // Wait (And around we go!)
        return;
    }

    // Boost on
```

```
if (PS3.getButtonClick(TRIANGLE)) {
    if(halfpowertimer==0){
        motormax=255;
        PS3.setRumbleOn(RumbleLow); // A DS3 will have fun here
        Serial.println(F("Boost! ON.));
        halfpowertimer=HALFPOWERTIMEOUT;
    }
}

// Manual Boost off
if (PS3.getButtonClick(CROSS)) {
    motormax=HALFPOWER;
    if(halfpowertimer>0){ // Reward turning off boost before timeout
        halfpowertimer=-((HALFPOWERTIMEOUT-halfpowertimer)*2); // Timer is
twice what remaINs on the timer
//    halfpowertimer=HALFPOWERCOODLDOWN; // Disable timer
    }
//    halfpowertimer=0; // Uncomment to disable cooldown timer
    PS3.setRumbleOff();
    Serial.println(F("Boost! OFF.));
}

// Boost timeout
if(halfpowertimer==1){
    halfpowertimer=HALFPOWERCOODLDOWN; // Disable timer
    halfpowertimer=halfpowertimer*2;
    motormax=HALFPOWER;
    PS3.setRumbleOff();
    Serial.println(F("Boost! AUTO OFF.));
}else if(halfpowertimer>0){ // Boost timeout
    halfpowertimer--;
}else if(halfpowertimer<0){ // Cooldown timeout
    halfpowertimer++;
}

// Select control mode and display on PS3's LEDs
if (PS3.getButtonClick(SELECT)) {
    controlmode++;
    if(controlmode>4){ controlmode=0; }
    PS3.setLedOff();
    switch (controlmode){
        case 0:
            break;
        case 1:
            PS3.setLedOn(LED1);
            break;
        case 2:
            PS3.setLedOn(LED2);
```

```
        break;
        case 3:
            PS3.setLedOn(LED3);
            break;
        case 4:
            PS3.setLedOn(LED4);
    }
}

// Servo weapon stuff.
// And 'aux' stuff
    if (PS3.getButtonPress(UP)) {    // While button is being pressed, allow
spin up to happen slowly
        weppval=WEPPMAX;
    } else if(PS3.getButtonPress (DOWN)) {    // Spin down fast
        weppval=0;
        webb_motor=0;
    } else {
        weppval=0;    // Spin down slow
    }

    // Spin weapons up towards full speed and down as needed to prevent
gearbox damage
    if(wepptimer<loopmills){    // Timer just for updating this
        if(webb_motor<weppval){
            webb_motor=webb_motor+WEPPRATEDELTA;
        } else if(webb_motor>0) {
            webb_motor=webb_motor-WEPPRATEDELTA;
        }
        wepptimer=loopmills+WEPPTIMERVALUE;    // reset timer
    }
// servo aux goes here!

// end

// Input debugging, also check battery voltage
if(debugtimer<loopmills){
    // Divider network:  GND---4.6k---A0---8.2k---Vbatt
    // Values: 9v = 671,  9.5v=712, 11.1v=833, 12.6v=941
    Serial.print(F("V:"));
    battvolt= analogRead(VINPIN);
    battvolt = map(battvolt, 671, 941, 0,254 );
    Serial.print(battvolt);
//    battvolt = 20;    // Override for hardware testing.    ** COMMENT OUT
THIS LINE BEFORE USE **
    // Battery too low, disable motors, flash first neopixel red
```

```
    if(battvolt<10){ // Red flash
//    while (PS3.getButtonClick(TRIANGLE)){ // <-- What is this for?
(Some old debug code?)
        setPixelRGB(0, 00, 00, 00);
        setPixelRGB(1, 00, 00, 00);
        delay(200);
        setPixelRGB(0, 60, 00, 00);
        setPixelRGB(1, 00, 00, 00);
        motormax=0;
    } else if(battvolt<50) { // Battery just low: *Red*
        setPixelRGB(0, 60, 00, 00);

    } else if(battvolt<100) { // Battery OKish: *Orange*
        setPixelRGB(0, 60, 30, 00);
    } else { // Battery is fine! So: *Green*
        setPixelRGB(0, 00, 30, 00);
    }
    Serial.print(F(" M:"));
    Serial.print(controlmode,DEC); // Why does this not display!?!?
    switch (controlmode){
        case 0:
            Serial.print(F(" RHx:"));
            Serial.print(PS3.getAnalogHat(RightHatX));
            Serial.print(F(" RHy: "));
            Serial.print(PS3.getAnalogHat(RightHatY));
            Serial.print(F(" LHy:"));
            Serial.print(PS3.getAnalogHat(LeftHatY));
            Serial.print(F(" LHx:"));
            Serial.print(PS3.getAnalogHat(LeftHatX));
            Serial.print(F(" L2:"));
            Serial.print(PS3.getAnalogButton(L2));
            Serial.print(F(" R2:"));
            Serial.print(PS3.getAnalogButton(R2));
            Serial.print(F("\t"));
            break;
        case 1:
            Serial.print(F(" RHx:"));
            Serial.print(PS3.getAnalogHat(RightHatX));
            Serial.print(F("\tRHy: "));
            Serial.print(PS3.getAnalogHat(RightHatY));
            break;
        case 2:
            Serial.print(F(" LHy:"));
            Serial.print(PS3.getAnalogHat(LeftHatY));
            Serial.print(F(" RHx:"));
            Serial.print(PS3.getAnalogHat(RightHatX));
            break;
        case 3:
```



```
        break;
    case 4:
        Serial.print(F(" LHx:"));
        Serial.print(PS3.getAnalogHat(LeftHatX));
        Serial.print(F(" L2:"));
        Serial.print(PS3.getAnalogButton(L2));
        Serial.print(F(" R2:"));
        Serial.print(PS3.getAnalogButton(R2));
        break;
    }
    Serial.print(F("\t "));
}

// Actually read in values from controller
// And bring things into pos/neg
switch (controlmode){
    case 1: // Default: 1=Right stick. Cent off, X=Steer. Y=Speed.
        dirval=PS3.getAnalogHat(RightHatX)-128;
        speedval=PS3.getAnalogHat(RightHatY)-128;
        // Invert axis if needed
        speedval = -speedval;
        // dirval =-dirval;
        break;
    case 2: // 2=Genric 2 stick: left stick Y axis Throttle. Right stick X
axis steering.
        dirval=PS3.getAnalogHat(RightHatX)-128;
        speedval=PS3.getAnalogHat(LeftHatY)-128;
        // Invert axis if needed
        speedval = -speedval;
        // dirval =-dirval;
        break;
    case 3: // 3=D-Pad
        if (PS3.getButtonClick(UP)) { }
        if (PS3.getButtonClick(RIGHT)) { }
        if (PS3.getButtonClick(DOWN)) { }
        if (PS3.getButtonClick(LEFT)) { }
        dirval=0;
        speedval=0;
        break;
    case 4: // 4=Hayden: left backward, Right forward, left stick X axis
steering.
        dirval=PS3.getAnalogHat(LeftHatX)-128;
        speedval=PS3.getAnalogButton(L2);
        speedval=speedval-PS3.getAnalogButton(R2);
        speedval = map(speedval, -255, 255, 128,-128 );
        // speedval = -speedval;
        // dirval =-dirval;
        break;
```

```
}

// Dead zone, to reduce twitching.
if(abs(dirval)<DEADZONE){
    dirval=0;
}
if(abs(speedval)<DEADZONE){
    speedval=0;
}

//  dirval = map(dirval, 128, -128, 50,-50 );

// What do we have in the regs?
if(debugtimer<loopmills){
    Serial.print("D:"); Serial.print(dirval); Serial.print("\t");
    Serial.print("S:"); Serial.print(speedval); Serial.print("\t");
    Serial.print("W:"); Serial.print(weppval); Serial.print("\t");
    Serial.print("A:"); Serial.print(auxval); Serial.print("\t");
}

switch (bottype){

    case 0:    // Test/debug
        break;

    case 1:    // Skidbot
/*
// Byron's janky Two channels to differential
if(speedval>1){    // Forward
    left_motor=speedval-dirval;
    right_motor=speedval+dirval;
}
else if(speedval<-1){    // Backward
    left_motor=speedval+dirval;
    right_motor=speedval-dirval;
}
else{    // Stopped
    left_motor=0;
    right_motor=0;
}
*/

//https://www.reddit.com/r/arduino/comments/49nltn/differential_steering_mix
ingformula/d0tdibk/
// Much better differential, bodged to work.
```

```
// Still not great at low-but-non-zero speeds
/*
if (speedval == 0 && dirval !=0) {
    left_motor = dirval;
    right_motor = -dirval;
} else {
    left_motor = speedval * ((-128 - dirval) / -128.0);
    right_motor = speedval * ((128 - dirval) / 128.0);

    if (speedval > 0 && left_motor > speedval)
        left_motor = speedval;
    if (speedval > 0 && right_motor > speedval)
        right_motor = speedval;
    if (speedval < 0 && left_motor < speedval)
        left_motor = speedval;
    if (speedval < 0 && right_motor < speedval)
        right_motor = speedval;
}
*/

/* Kell example:  Scale tune to be good.  Offsets should be zero
motor_r = scale*(forward - forward_offset + turn - turn_offset);
motor_l = scale*(-forward + forward_offset + turn - turn_offset);
*/
right_motor = (speedval + dirval );
left_motor = (-speedval + dirval );
break;
case 2:  // Steerbot - servo steering.
    left_motor = speedval;
    right_motor = dirval;
break;
}

// Show current motor staus on LEDs
motcurrmax=(abs(speedval)*1.55);    // Scale 0-128 to 0-255. I think.
if(motormax>HALFPOWER){ // Orange
    if(motcurrmax>113){ // In 'danger zone'
        setPixelRGB(1, motcurrmax+10, 248-(motcurrmax*1.25), 00);
    } else { // 'Boost' enabled but not in zone
        setPixelRGB(1, motcurrmax+10, motcurrmax+10, 00);
    }
} else{ // Green only
    setPixelRGB(1, 00, motcurrmax+10, 0);
}

// rescale from +/- 128 to +/- motormax, to allow for boost stuff
left_motor = map(left_motor, 128, -128, motormax,-motormax);
```

```
right_motor = map(right_motor, 128, -128, motormax,-motormax);

// Lets see what is going to the motors
if(debugtimer<loopmills){
  debugtimer=loopmills+DEBUGVALUE; // Last 'debug if': reset timer
  Serial.print("LED:"); Serial.print(motcurrmax); Serial.print("\t");
  Serial.print("L:"); Serial.print(left_motor); Serial.print("\t");
  Serial.print("R:"); Serial.print(right_motor); Serial.print("\t");
  Serial.print("W:"); Serial.print(webb_motor); Serial.print("\t");
  Serial.print("A:"); Serial.print(aux_motor); Serial.print("\t");
  Serial.print("B:"); Serial.print(halfpowertimer); Serial.print("\n");
}

// Different motor configs

switch (bottype){

  case 0: // Test/debug
    break;
  case 1: // Skidbot
    // Send to motors,
    setmotor(0,left_motor);
    setmotor(1,right_motor); // Oooh! This is why it was this way!
    // And servo(s)
    setservo(0,webb_motor);
    setservo(1,aux_motor);
    break;
  case 2: // Steerbot
    // Send to motors,
    setmotor(0,left_motor); // Actually 'forward'
    setmotor(1,webb_motor);
    // And servo(s)
    setservo(0,right_motor); // Odd but hey, this is how it's written
    setservo(1,aux_motor);
    break;
}

} // loop() end

// Set H bridge motor output controls and PWM speeds.
// 255 = full forward. -255 full reverse
void setmotor(char mot,int mot_speed){
  constrain(mot_speed,-255,255); // Prevent values from going out of range
  // Left motor
```

```
if(mot==0){
  if(mot_speed>0){          // Forward
    digitalWrite(aIN1, LOW);
    digitalWrite(aIN2, HIGH);
    analogWrite(pwmA, abs(mot_speed)); // PWM output for motor speed
  } else if(mot_speed<0) { // Backward
    digitalWrite(aIN1, HIGH);
    digitalWrite(aIN2, LOW);
    analogWrite(pwmA, abs(mot_speed)); // PWM output for motor speed
  } else { // Break/stop
    digitalWrite(aIN1, LOW);
    digitalWrite(aIN2, LOW);
    analogWrite(pwmA, 0);
  }
}
// Left motor
if(mot==1){
  if(mot_speed>0){          // Forward
    digitalWrite(bIN1, LOW);
    digitalWrite(bIN2, HIGH);
    analogWrite(pwmB, abs(mot_speed)); // PWM output for motor speed
  } else if(mot_speed<0) { // Backward
    digitalWrite(bIN1, HIGH);
    digitalWrite(bIN2, LOW);
    analogWrite(pwmB, abs(mot_speed)); // PWM output for motor speed
  } else { // Break/stop
    digitalWrite(bIN1, LOW);
    digitalWrite(bIN2, LOW);
    analogWrite(pwmB, 0);
  }
}
}

// Turn -128 128 range into servo output
void setservo(char mot,int servpos){
  constrain(servpos,-128,128); // Prevent values from going out of range
// servpos = map(servpos, -128, 128, 5,175 ); // And convert to servo
range
servpos = map(servpos, -128, 128, 50,130 ); // And convert to servo
range

  if(mot==0){
    servo_one.write(servpos);
  }
  if(mot==1){
    servo_two.write(servpos);
  }
}
```

```
}

}

// Motor test function,
void dcmot_test(){
    //Mot one
    setPixelRGB(0, 00, 30, 00); // Forward
    setmotor(0,128);
    delay(1000);
    setPixelRGB(0, 00, 00, 00);
    setmotor(0,0);
    delay(500);
    setPixelRGB(0, 30, 00, 00); //Backward
    setmotor(0,-128);
    delay(1000);
    setPixelRGB(0, 00, 00, 00);
    setmotor(0,0);
    delay(1000);
    // Mot two
    setPixelRGB(1, 00, 30, 00);
    setmotor(1,128);
    delay(1000);
    setPixelRGB(1, 00, 00, 00);
    setmotor(1,0);
    delay(500);
    setPixelRGB(1, 30, 00, 00);
    setmotor(1,-128);
    delay(1000);
    setPixelRGB(1, 00, 00, 00);
    setmotor(1,0);

    delay(1000);
}

// Sets RGB leds to current RGB value, regardless of what library
void setPixelRGB(int Pixel, byte red, byte green, byte blue) {
    // light
    cRGB value;
    value.b = blue; value.g = green; value.r = red; // RGB Value
    LED.set_crgb_at(Pixel, value); // Set value at LED found at index 0
    LED.sync(); // Sends the value to the LED
    // Neopixel
    // strip.setPixelColor(Pixel, strip.Color(red, green, blue));
    // neopixel.show();
}
```