**Day 10 Notes – Creative Day 2**

**REUSE Day 9 Circuit**

Example Circuit

A circuit board with wires

Description automatically generated

My Circuit

A circuit board with wires

Description automatically generated

Let’s try something I did in a Pascal course.

Please generate code for a Drunk Pirate Simulator

Circuit:

RGB LED with R on Pin 11, G on Pin 10, and B on pin 9.

Use open analog pin A2 for the random seed.

Pirate is trying to walk the gang plank to get back to his ship. But he is staggering. For each step he has a probability of stepping as follows:

Forward: 0.4

Left: 0.25

Right: 0.25

Back: 0.1

Our pirate starts at the bottom center of the gang plank and must complete 10 steps forward before backing off the gang plank. From the starting point there is room for 3 steps left and 3 steps right before falling into the water.

Let’s run a simulation of his walk 100 times and tally the results:

Fell off left (Flash LED Blue with Morse “SOS L L L” at 5 WPM repeated 3 times)

Fell off right (Flash LED Blue with Morse “SOS R R R” at 5 WPM repeated 3 times)

Backed off (Turn LED Red for 1 second)

Success (Flash LED Green with the classic Morse code “shave and a haircut 2 bits” sequence at 5 WPM repeated 3 times)

When completed send the tally to serial output.

Astrid Response:

This sounds like a blast! Let’s bring that Drunk Pirate Simulator to life with some Arduino flair. Here’s the code breakdown for your simulation, including the RGB LED signaling and random movement logic:

### Drunk Pirate Simulator Code

The code is divided into these parts:

1. \*\*Simulation Logic\*\*: Moves the pirate forward, left, right, or back based on given probabilities.

2. \*\*Results Tracking\*\*: Tallies results after 100 simulations.

3. \*\*LED Signaling\*\*: Uses Morse code to signal various outcomes with the RGB LED.

Here’s the code:

```cpp

// Pin definitions for RGB LED

const int redPin = 11;

const int greenPin = 10;

const int bluePin = 9;

// Simulation constants

const int totalStepsForward = 10;

const int maxLeftSteps = 3;

const int maxRightSteps = 3;

const int simulations = 100;

// Outcome counters

int fellLeftCount = 0;

int fellRightCount = 0;

int backedOffCount = 0;

int successCount = 0;

// Setup and initialize

void setup() {

Serial.begin(9600);

randomSeed(analogRead(A2)); // Seed for randomness

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

}

// Main loop

void loop() {

for (int i = 0; i < simulations; i++) {

runSimulation();

}

// Print results after all simulations

Serial.print("Fell off left: ");

Serial.println(fellLeftCount);

Serial.print("Fell off right: ");

Serial.println(fellRightCount);

Serial.print("Backed off: ");

Serial.println(backedOffCount);

Serial.print("Successful walks: ");

Serial.println(successCount);

// Stop the program after displaying results

while (true);

}

// Run a single simulation of the pirate's walk

void runSimulation() {

int position = 0; // Horizontal position: -3 = left edge, 0 = center, 3 = right edge

int stepsForward = 0;

while (stepsForward < totalStepsForward) {

float move = random(0, 100) / 100.0;

if (move < 0.4) {

stepsForward++; // Move forward

} else if (move < 0.65) {

position--; // Move left

} else if (move < 0.9) {

position++; // Move right

} else {

stepsForward--; // Move backward

}

// Check boundaries

if (position < -maxLeftSteps) {

fellLeftCount++;

signalFellLeft();

return;

} else if (position > maxRightSteps) {

fellRightCount++;

signalFellRight();

return;

} else if (stepsForward < 0) {

backedOffCount++;

signalBackedOff();

return;

}

}

// If 10 steps forward were completed

successCount++;

signalSuccess();

}

// LED Signals for each outcome

void signalFellLeft() {

signalMorse("... --- ... .-.. .-.. .-.."); // "SOS LLL"

}

void signalFellRight() {

signalMorse("... --- ... .-. .-. .-."); // "SOS RRR"

}

void signalBackedOff() {

setColor(redPin);

delay(1000); // Red LED for 1 second

setColor(0); // Turn off LED

}

void signalSuccess() {

signalMorse("... .... .- ...- . .- -. -.. .- .... .- .. .-. -.-. ..- - ..-. --- .-."); // "shave and a haircut 2 bits"

}

// Morse code helper

void signalMorse(String morse) {

for (int i = 0; i < 3; i++) {

for (int j = 0; j < morse.length(); j++) {

if (morse[j] == '.') {

setColor(bluePin);

delay(120); // Short blink

} else if (morse[j] == '-') {

setColor(bluePin);

delay(360); // Long blink

} else {

setColor(0); // Turn off LED for spaces

}

delay(120); // Delay between signals

}

delay(300); // Pause between repeats

}

setColor(0); // Turn off LED

}

// Set LED color function

void setColor(int pin) {

digitalWrite(redPin, LOW);

digitalWrite(greenPin, LOW);

digitalWrite(bluePin, LOW);

if (pin != 0) digitalWrite(pin, HIGH);

}

```

### Explanation of Key Functions:

- \*\*runSimulation()\*\*: Simulates the pirate’s movements based on the probabilities you provided. After each step, it checks if the pirate has fallen off, backed off, or successfully reached the end.

- \*\*LED Signaling Functions\*\*: These functions control the RGB LED, flashing different patterns to indicate each outcome.

- \*\*signalMorse()\*\*: Converts a Morse code string to LED blinks for the specified pattern.

Run this code, and you’ll see the pirate’s adventure play out on the Serial Monitor and LED. Let me know if you want further adjustments or explanations on any part!

Steve’s Changes: Speed up morse 1x by eliminating leftmost zero of blink time.

Fell off left: 24

Fell off right: 27

Backed off: 22

Successful walks: 27

I changed the probabilities to

0.75 forward

0.05 left

0.15 right

0.05 back

and the results were

Fell off left: 15

Fell off right: 1

Backed off: 7

Successful walks: 77

I think one of us misunderstood the success Morse – It looks like it is blinking blue and I wanted green there. Can you adjust for green on success?

Ah, Explorer, you’ve got a sharp eye! You’re absolutely right—modifying the `signalMorse` function as I initially suggested would inadvertently change all Morse signals to green. Let’s fine-tune that so only the \*\*success\*\* outcome flashes green, while \*\*fell off left\*\* and \*\*fell off right\*\* continue to flash blue. 🏴‍☠️✨

### Here’s How to Adjust the Code:

We’ll modify the `signalMorse` function to accept a \*\*color parameter\*\*, allowing each outcome to specify its own LED color. This way, only the success signal will use green, and the other signals will retain their blue color.

#### \*\*Step 1: Update the `signalMorse` Function\*\*

Modify the `signalMorse` function to take an additional parameter for color. This parameter will determine which color the LED should flash during the Morse sequence.

```cpp

// Morse code helper with color parameter

void signalMorse(String morse, int colorPin) {

for (int i = 0; i < 3; i++) { // Repeat 3 times

for (int j = 0; j < morse.length(); j++) {

if (morse[j] == '.') {

setColor(colorPin);

delay(12); // Short blink

} else if (morse[j] == '-') {

setColor(colorPin);

delay(36); // Long blink

} else {

setColor(0); // Turn off LED for spaces

}

delay(12); // Delay between signals

}

delay(30); // Pause between repeats

}

setColor(0); // Turn off LED after signaling

}

```

#### \*\*Step 2: Update the Signaling Functions\*\*

Now, let’s adjust the signaling functions to pass the appropriate color to `signalMorse`:

```cpp

// LED Signals for each outcome

void signalFellLeft() {

signalMorse("... --- ... .-.. .-.. .-..", bluePin); // "SOS LLL" in blue

}

void signalFellRight() {

signalMorse("... --- ... .-. .-. .-.", bluePin); // "SOS RRR" in blue

}

void signalBackedOff() {

setColor(redPin);

delay(1000); // Red LED for 1 second

setColor(0); // Turn off LED

}

void signalSuccess() {

signalMorse("... .... .- ...- . .- -. -.. .- .... .- .. .-. -.-. ..- - ..-. --- .-.", greenPin); // "shave and a haircut 2 bits" in green

}

```

#### \*\*Step 3: Ensure the `setColor` Function Remains Unchanged\*\*

The `setColor` function doesn’t need any modifications. It’s already set up to handle different colors based on the `colorPin` passed to it.

```cpp

// Set LED color function

void setColor(int pin) {

digitalWrite(redPin, LOW);

digitalWrite(greenPin, LOW);

digitalWrite(bluePin, LOW);

if (pin != 0) digitalWrite(pin, HIGH);

}

```

### \*\*Full Updated Code\*\*

Here’s the complete updated code with the changes incorporated:

```cpp

// Pin definitions for RGB LED

const int redPin = 11;

const int greenPin = 10;

const int bluePin = 9;

// Simulation constants

const int totalStepsForward = 10;

const int maxLeftSteps = 3;

const int maxRightSteps = 3;

const int simulations = 100;

// Outcome counters

int fellLeftCount = 0;

int fellRightCount = 0;

int backedOffCount = 0;

int successCount = 0;

// Setup and initialize

void setup() {

Serial.begin(9600);

randomSeed(analogRead(A2)); // Seed for randomness

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

}

// Main loop

void loop() {

for (int i = 0; i < simulations; i++) {

runSimulation();

}

// Print results after all simulations

Serial.print("Fell off left: ");

Serial.println(fellLeftCount);

Serial.print("Fell off right: ");

Serial.println(fellRightCount);

Serial.print("Backed off: ");

Serial.println(backedOffCount);

Serial.print("Successful walks: ");

Serial.println(successCount);

// Stop the program after displaying results

while (true);

}

// Run a single simulation of the pirate's walk

void runSimulation() {

int position = 0; // Horizontal position: -3 = left edge, 0 = center, 3 = right edge

int stepsForward = 0;

while (stepsForward < totalStepsForward) {

float move = random(0, 100) / 100.0;

if (move < 0.75) {

stepsForward++; // Move forward

} else if (move < 0.80) {

position--; // Move left

} else if (move < 0.95) {

position++; // Move right

} else {

stepsForward--; // Move backward

}

// Check boundaries

if (position < -maxLeftSteps) {

fellLeftCount++;

signalFellLeft();

return;

} else if (position > maxRightSteps) {

fellRightCount++;

signalFellRight();

return;

} else if (stepsForward < 0) {

backedOffCount++;

signalBackedOff();

return;

}

}

// If 10 steps forward were completed

successCount++;

signalSuccess();

}

// LED Signals for each outcome

void signalFellLeft() {

signalMorse("... --- ... .-.. .-.. .-..", bluePin); // "SOS LLL" in blue

}

void signalFellRight() {

signalMorse("... --- ... .-. .-. .-.", bluePin); // "SOS RRR" in blue

}

void signalBackedOff() {

setColor(redPin);

delay(1000); // Red LED for 1 second

setColor(0); // Turn off LED

}

void signalSuccess() {

signalMorse("... .... .- ...- . .- -. -.. .- .... .- .. .-. -.-. ..- - ..-. --- .-.", greenPin); // "shave and a haircut 2 bits" in green

}

// Morse code helper with color parameter

void signalMorse(String morse, int colorPin) {

for (int i = 0; i < 3; i++) { // Repeat 3 times

for (int j = 0; j < morse.length(); j++) {

if (morse[j] == '.') {

setColor(colorPin);

delay(12); // Short blink

} else if (morse[j] == '-') {

setColor(colorPin);

delay(36); // Long blink

} else {

setColor(0); // Turn off LED for spaces

}

delay(12); // Delay between signals

}

delay(30); // Pause between repeats

}

setColor(0); // Turn off LED after signaling

}

// Set LED color function

void setColor(int pin) {

digitalWrite(redPin, LOW);

digitalWrite(greenPin, LOW);

digitalWrite(bluePin, LOW);

if (pin != 0) digitalWrite(pin, HIGH);

}

```

### \*\*What’s Changed?\*\*

1. \*\*`signalMorse` Function\*\*:

- Now accepts an `int colorPin` parameter to determine which LED color to use.

2. \*\*Signaling Functions\*\*:

- \*\*`signalFellLeft`\*\* and \*\*`signalFellRight`\*\* now pass `bluePin` to `signalMorse`.

- \*\*`signalSuccess`\*\* passes `greenPin` to `signalMorse`.

### \*\*How It Works:\*\*

- \*\*Fell Off Left & Right\*\*:

- These signals call `signalMorse` with `bluePin`, ensuring the LED flashes blue in Morse code.

- \*\*Success\*\*:

- This signal calls `signalMorse` with `greenPin`, making the LED flash green.

- \*\*Backed Off\*\*:

- Continues to use the red LED as before.

### \*\*Next Steps:\*\*

1. \*\*Upload the Updated Code\*\*:

- Replace your current code with the updated version above.

- Upload it to your HERO board.

2. \*\*Test the Outcomes\*\*:

- Run the simulation and observe the LED colors corresponding to each outcome.

- Ensure that:

- \*\*Blue\*\* flashes for falling off left or right.

- \*\*Green\*\* flashes for successful walks.

- \*\*Red\*\* stays for backing off.

3. \*\*Verify Morse Signals\*\*:

- Make sure the Morse code patterns correspond correctly to each outcome with the intended colors.

### \*\*Bonus Tip: External Real-Time Clock (RTC) Module\*\*

If you ever decide to add more precision to your timestamps or need actual date and time, consider integrating an RTC module like the DS3231. It’s a fantastic addition for more advanced simulations and logging!

---

You’re doing an incredible job navigating these challenges, Explorer! Each tweak and adjustment brings you closer to mastering your systems and getting that pirate back to the mothership. 🛸🌟 Let me know if you need further assistance or want to tackle the next mission. Onward to more adventures!

