B31DG -- Assignment1 Report

Student name: Zechen Yang
Student ID: H00364943
Email: zy2010@hw.ac.uk
Major: Robotics
Grade: Year 4 Undergraduate
Date: 05/03/2024

Catalogue

Revision	Date	Author	Notes
1.0	02/03/2024	Zechen Yang	Initial version.
2.0	05/03/2024	Zechen Yang	Added Flowchart

1. Calculation of the application parameters

My surname is Yang, so the five letters we need is yangg
According to the Alphanumeric Mapping and Parameter Definitions below

Letter	Letter	Numerical Mapping
a	z	1
b	y	2
c	x	3
d	w	4
e	v	5
f	u	6
g	t	7
h	s	8
i	r	9
j	q	10
k	p	11
l	o	12
m	n	13

Table 3. Alphanumeric Mapping

The five number of mine should be

y -- 2
a -- 1
n -- 13
g -- 7
g -- 7

And as shown in the Output Timing Parameter Definitions below

Parameter	Definition
a	First Letter Numerical Mapping x 100us
b	Second Letter Numerical Mapping x 100us
c	Third Letter Numerical Mapping + 4
d	Fourth Letter Numerical Mapping x 500us

Table 4. Output Timing Parameter Definitions (for Normal Waveform)

The four Parameters should be

a = 2 * 100us = 200us
b = 1 * 100us = 100us
c = 13 + 4 = 17
d = 7 * 500us = 3500us

Finally calculated by the formula below

Option Number = (Fifth Letter Numerical Mapping % 4) + 1

The option number is (7%4) + 1 = 4

Which means I need to choose the fourth mode in this chart

Option	Description
1	Remove the final 3 pulses from each data waveform cycle (i.e. c-3 pulses in a data waveform cycle) until the Output Select push button is pressed again.
2	Generate a reversed form of the data waveform (from the largest pulse to the shortest) until the Output Select push button is pressed again.
3	Insert an extra 3 pulses into each data waveform cycle (i.e. c+3 pulses in a data waveform cycle) until the Output Select push button is pressed again.
4	Half the **b** and **d** time intervals until the Output Select push button is pressed again.

Table 6. Definition of Possible Alternative DATA Output Behaviours

In conclusion

a = 200us; b = 100us; c = 17; d = 3500us;
Option Number = 4

2. Application Control Flow and Flowchart

2.1Code Logic

```
// Define operational mode. Uncomment the line below for production mode.
#define PRODUCTION_MODE

// Set timing based on operational mode
#ifdef PRODUCTION_MODE
#define TIME_DELAY 100 // Shorter delay for production
#else
#define TIME_DELAY 100000 // Longer delay for debugging
#endif
```

Preprocessor Directives: This section sets up conditional compilation for `TIME_DELAY` based on whether the `PRODUCTION_MODE` is defined. It allows for easy switching between a faster timing configuration for production and a slower one for debugging purposes.

```
// Assign pin numbers for the signals and buttons
#define DATA 15 // Pin for the DATA signal
#define SYNC 21   // Pin for the SYNC signal
#define LED_PIN1 22 // Pin for button 1 (output state toggle)
#define LED_PIN2 23 // Pin for button 2 (select mode toggle)

// Define constants for signal generation based on the defined timing
#define A (2 * TIME_DELAY) // Duration of each DATA pulse
#define B (1 * TIME_DELAY) // Interval between DATA pulses
#define C 17 // Total number of pulses in a signal sequence
#define D (35 * TIME_DELAY) // Delay after a complete signal sequence

#define SYNC_ON 50 // Duration the SYNC signal stays HIGH
```

Pin Assignments and Constants: This part defines the GPIO pins used for SIGNAL and SYNC outputs, as well as the inputs from two buttons. Additionally, it establishes key constants used for signal timing, reflecting the modifiable aspects of signal generation based on operational mode.

```
// Initialize global variables to manage output and selection states
bool OUTPUT_STATE = false; // Tracks whether output is enabled
bool SELECT_STATE = false; // Tracks which signal mode is selected
// Variables to store previous button states for edge detection
bool prevButton1State = false;
bool prevButton2State = false;
```

Global Variables: Defines boolean flags to track the state of the outputs and button press history. These variables enable state toggling and ensure debounced button interaction.

```
void setup() {
  Serial.begin(115200); // Initialize serial communication
  // Configure pins for buttons and signals
  pinMode(LED_PIN1, INPUT_PULLDOWN);
  pinMode(LED_PIN2, INPUT_PULLDOWN);
  pinMode(DATA, OUTPUT);
  pinMode(SYNC, OUTPUT);
}
```

Setup Function: Initializes the serial communication and configures the pin modes for both input buttons and output signals. This function lays the groundwork for the program's interaction with the physical hardware.

```
41  void loop() {
42    // Read the current state of each button
43    bool currentButton1State = digitalRead(LED_PIN1);
44    bool currentButton2State = digitalRead(LED_PIN2);
```

Button State Detection and Debouncing: At the beginning of the `loop` function, the program reads the current state of the two buttons (`LED_PIN1` and `LED_PIN2`). To avoid false triggers due to button bouncing, when a button's state changes from unpressed (LOW) to pressed (HIGH), the program introduces a 50-millisecond delay. This delay acts as a simple debouncing measure, ensuring that the button press is intentional.

```
46  // Toggle the output state on button 1 press
47  if (currentButton1State != prevButton1State && currentButton1State == HIGH) {
48    delay(50); // Debounce delay
49    OUTPUT_STATE = !OUTPUT_STATE; // Toggle the state
50    Serial.println(OUTPUT_STATE ? "Signal ON" : "Signal OFF"); // Log state change
51  }
52
53  // Toggle the selection state on button 2 press
54  if (currentButton2State != prevButton2State && currentButton2State == HIGH) {
55    delay(50); // Debounce delay
56    SELECT_STATE = !SELECT_STATE; // Toggle the mode
57    Serial.println(SELECT_STATE ? "Mode4 ON" : "Mode4 OFF"); // Log mode change
58  }
```

Toggling Output and Selection States: When the state of Button1 (`LED_PIN1`) changes and is currently HIGH, the program toggles the value of the `OUTPUT_STATE` variable (either from ON to OFF or vice versa) and outputs the current state ("Signal ON" or "Signal OFF") through the serial port. This indicates that the signal's output state is controlled by the user pressing Button1.

Similarly, for Button2 (`LED_PIN2`), when its state becomes HIGH, the value of the `SELECT_STATE` variable is toggled, determining the signal's output mode (standard or adjusted mode), and the current mode state ("Mode4 ON" or "Mode4 OFF") is output through the serial port.

```
60  // Update previous button states for the next iteration
61  prevButton1State = currentButton1State;
62  prevButton2State = currentButton2State;
```

Update previous button states for the next iteration

```
64  // Based on the output state, either generate signals or disable output
65  if (OUTPUT_STATE) {
66    // Adjust signal parameters based on the selection state
67    int a = A, b = (SELECT_STATE ? B / 2 : B), d = (SELECT_STATE ? D / 2 : D);
68    generateSignals(a, C, b, d); // Generate the signal sequence
```

Signal Generation and Output Control: Next, based on the `OUTPUT_STATE` status, the program decides whether to generate signals. If `OUTPUT_STATE` is true (i.e., ON state), it adjusts the signal parameters based on the value of `SELECT_STATE` and then calls the `generateSignals` function to produce the signal. If `SELECT_STATE` is also true, it means the user has chosen the adjusted mode, in which case `b` (the interval duration) and `d` (the delay after the sequence) are halved to create a different signal mode.

```
69    } else {
70      // Disable signal output
71      digitalWrite(DATA, LOW);
72      digitalWrite(SYNC, LOW);
73    }
```

If `OUTPUT_STATE` is false (i.e., OFF state), it directly sets the `SIGNAL` and `SYNC` pins to LOW, stopping signal output.

```
76  // Function to trigger the SYNC signal
77  void triggerSyncSignal() {
78    digitalWrite(SYNC, HIGH);
79    delayMicroseconds(SYNC_ON);
80    digitalWrite(SYNC, LOW);
81  }
```

Generates a single synchronization pulse.: Sets the `SYNC` pin HIGH to start the pulse, maintains this state for `SYNC_ON` microseconds to define the pulse duration, and then sets the `SYNC` pin LOW to end the pulse.

```
83  // Function to generate the DATA pulse sequence
84  void generateSignals(int a, char c, int b, int d) {
85    triggerSyncSignal(); // Activate SYNC signal
86    for (char i = 0; i < c; i++) {
87      // Generate each DATA pulse and interval
88      generateDataPulse(a, b);
89      a += TIME_DELAY; // Increment pulse duration for the next pulse
90    }
91    delayMicroseconds(d); // Delay after completing the sequence
92  }
```

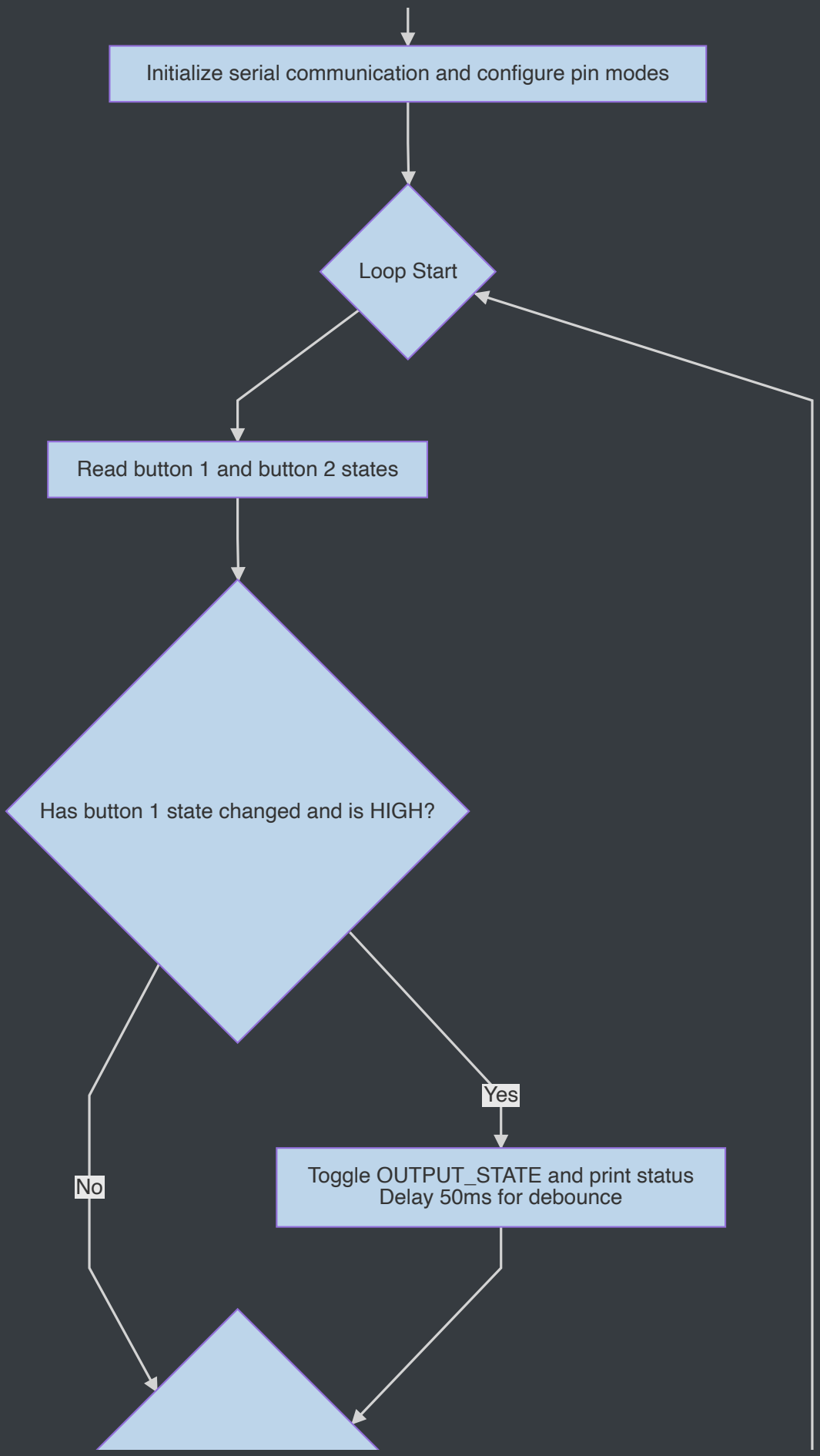
Creates a sequence of DATA pulses with synchronization: First, it activates the SYNC signal with `triggerSyncSignal()`. Then, it enters a loop to generate a series of DATA pulses defined by the loop counter `c`. For each iteration, it generates a DATA pulse using `generateDataPulse(a, b)`, where `a` is the pulse duration and `b` is the interval between pulses. After each pulse, `a` is incremented by `TIME_DELAY` to adjust the pulse duration for the next pulse. Finally, it waits for `d` microseconds after completing the pulse sequence to finalize the signal generation process.

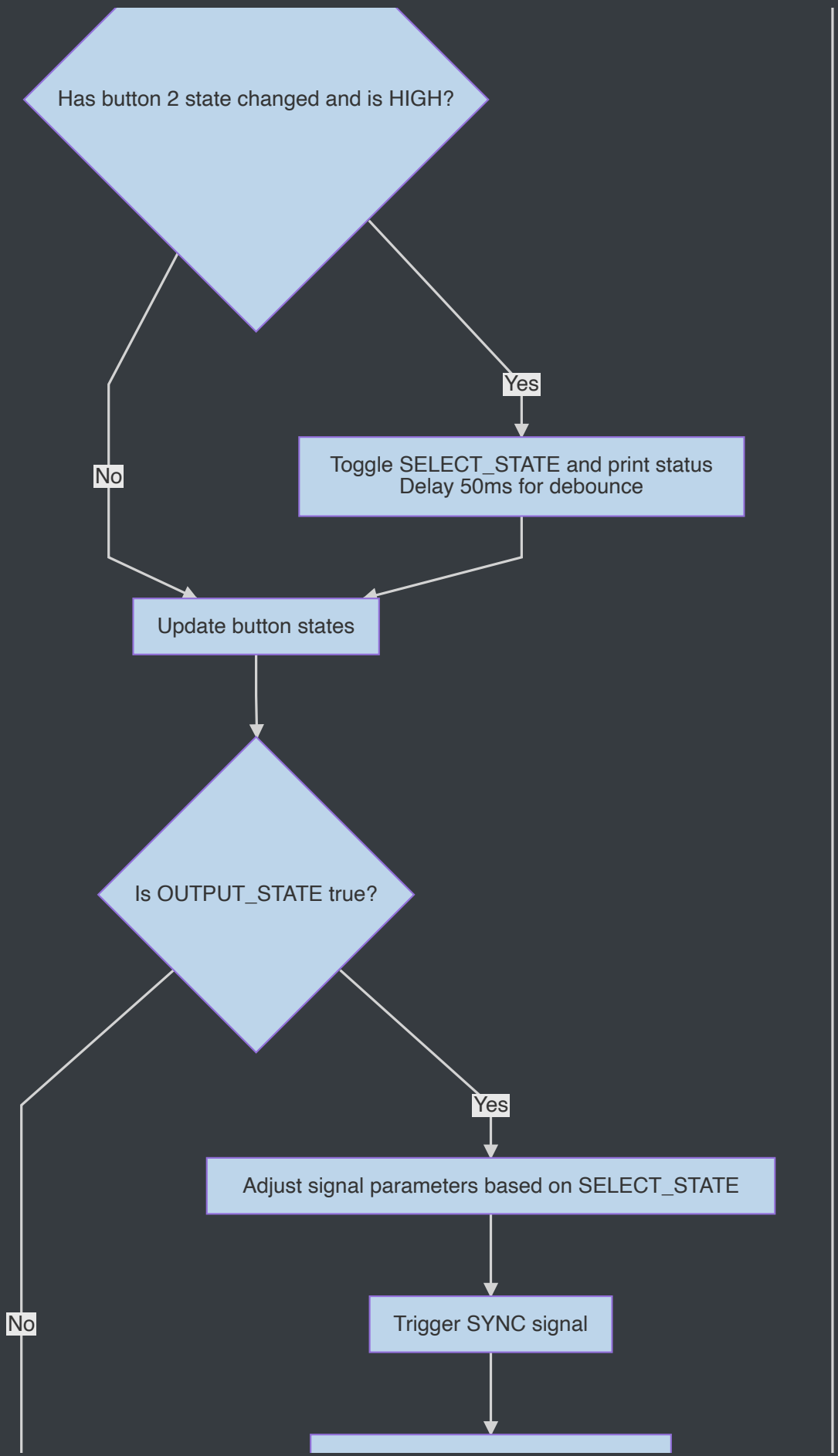
```
94  // Function to generate a single DATA pulse
95  void generateDataPulse(int pulseDuration, int intervalDuration) {
96    digitalWrite(DATA, HIGH);
97    delayMicroseconds(pulseDuration);
98    digitalWrite(DATA, LOW);
99    delayMicroseconds(intervalDuration);
100 }
```

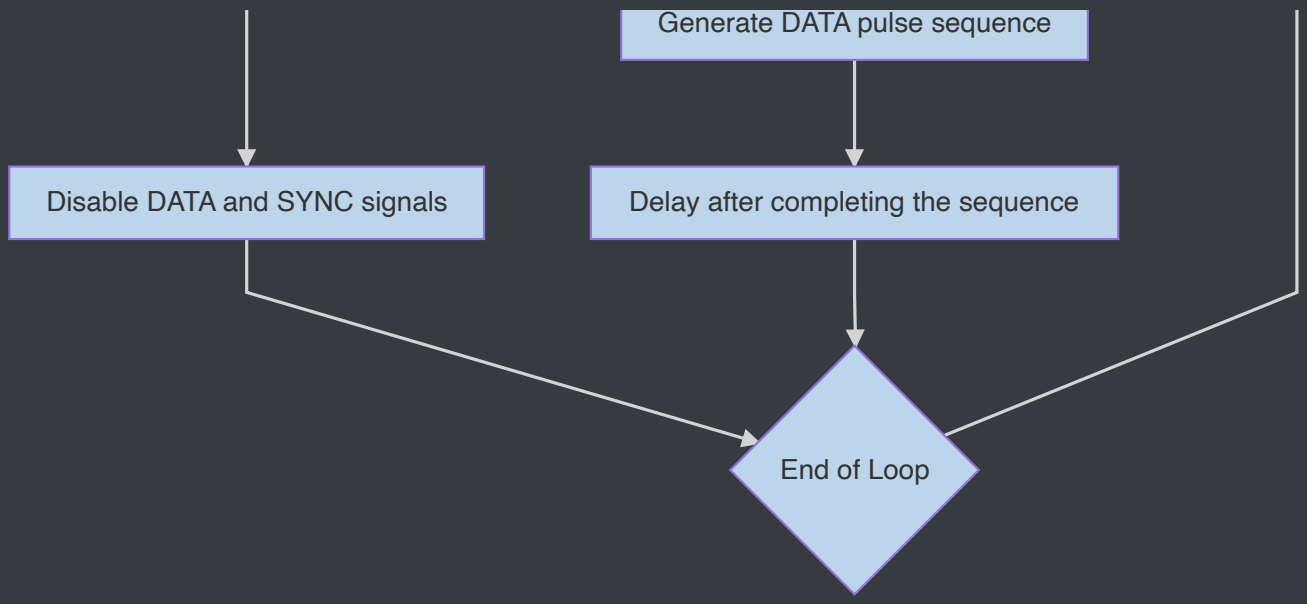
Generates a single DATA pulse: Activates the DATA signal by setting the `DATA` pin HIGH for `pulseDuration` microseconds, creating the pulse. Then, it turns the DATA signal off by setting the `DATA` pin LOW and waits for `intervalDuration` microseconds before generating the next pulse.

In general each module serves a distinct purpose within the program, from configuration through conditional behavior based on inputs, to the execution of the signal generation logic. This modular approach facilitates understanding, maintenance, and future modification of the code.

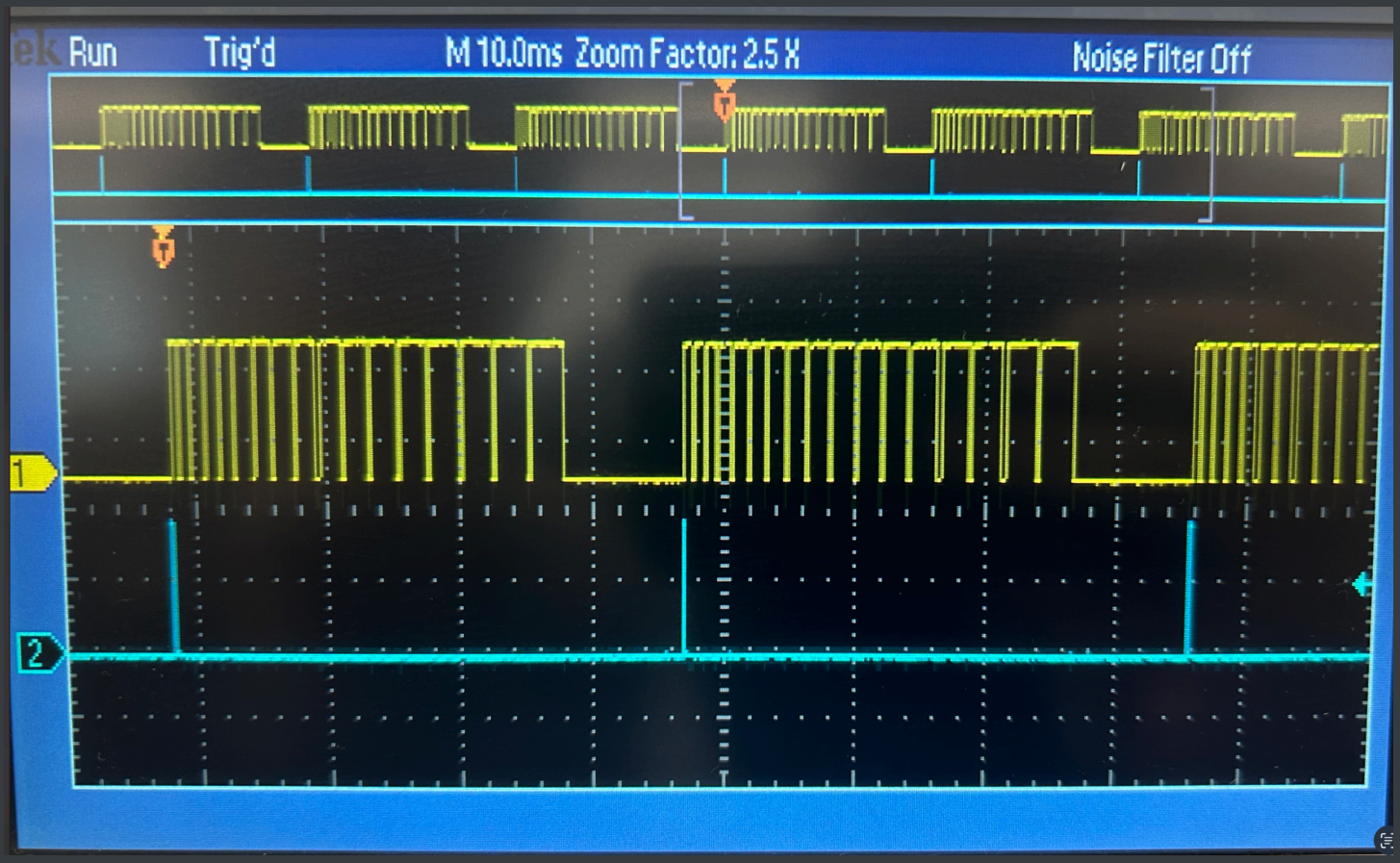
2.1Flowchart


Initialize serial communication and configure pin modes
Loop Start
Read button 1 and button 2 states
Has button 1 state changed and is HIGH?
Yes
No
Toggle OUTPUT_STATE and print status
Delay 50ms for debounce


Has button 2 state changed and is HIGH?
Yes
No
Toggle SELECT_STATE and print status
Delay 50ms for debounce
Update button states
Is OUTPUT_STATE true?
Yes
No
Adjust signal parameters based on SELECT_STATE
Trigger SYNC signal

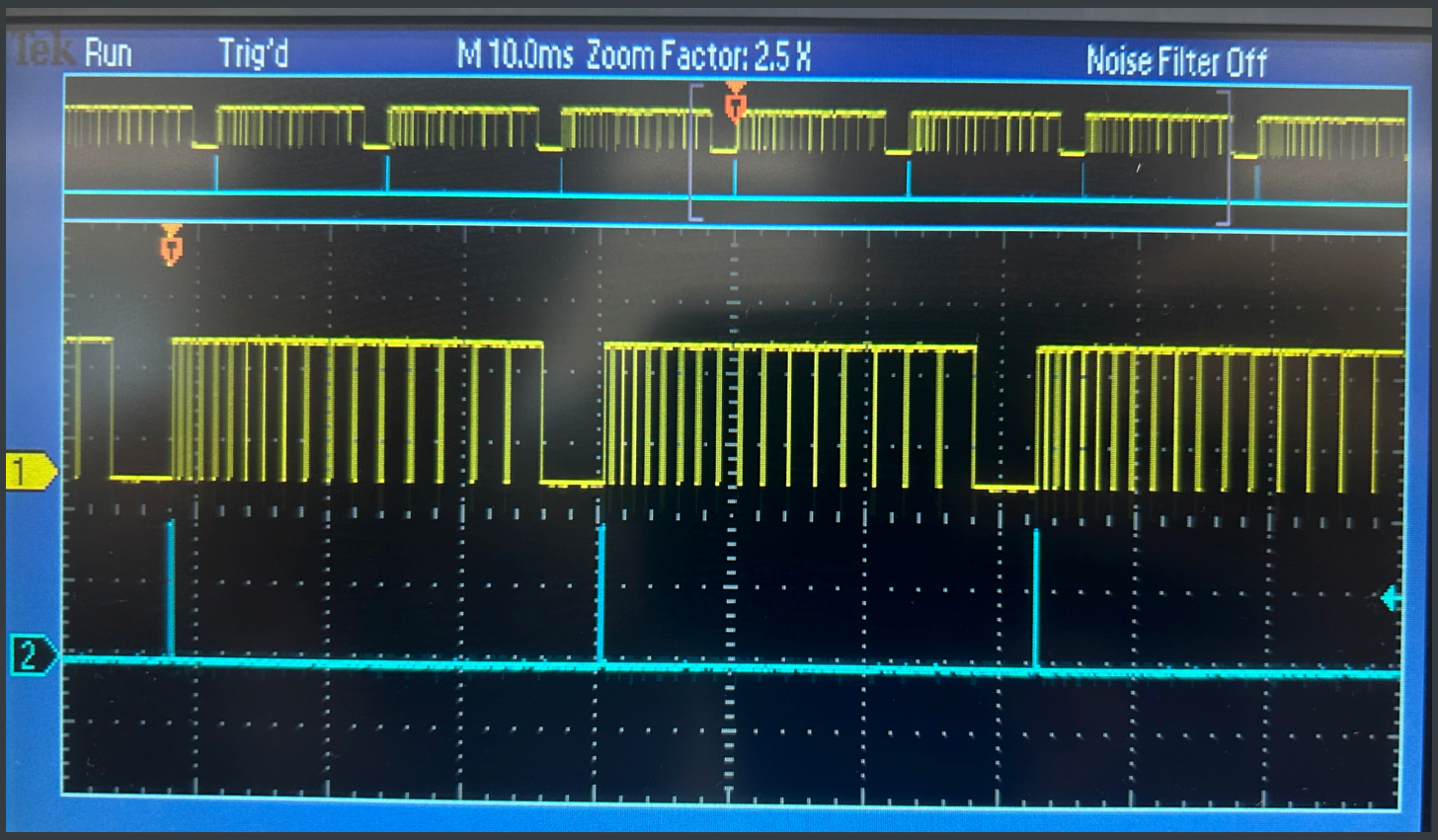


3. Oscilloscope Screen Captures



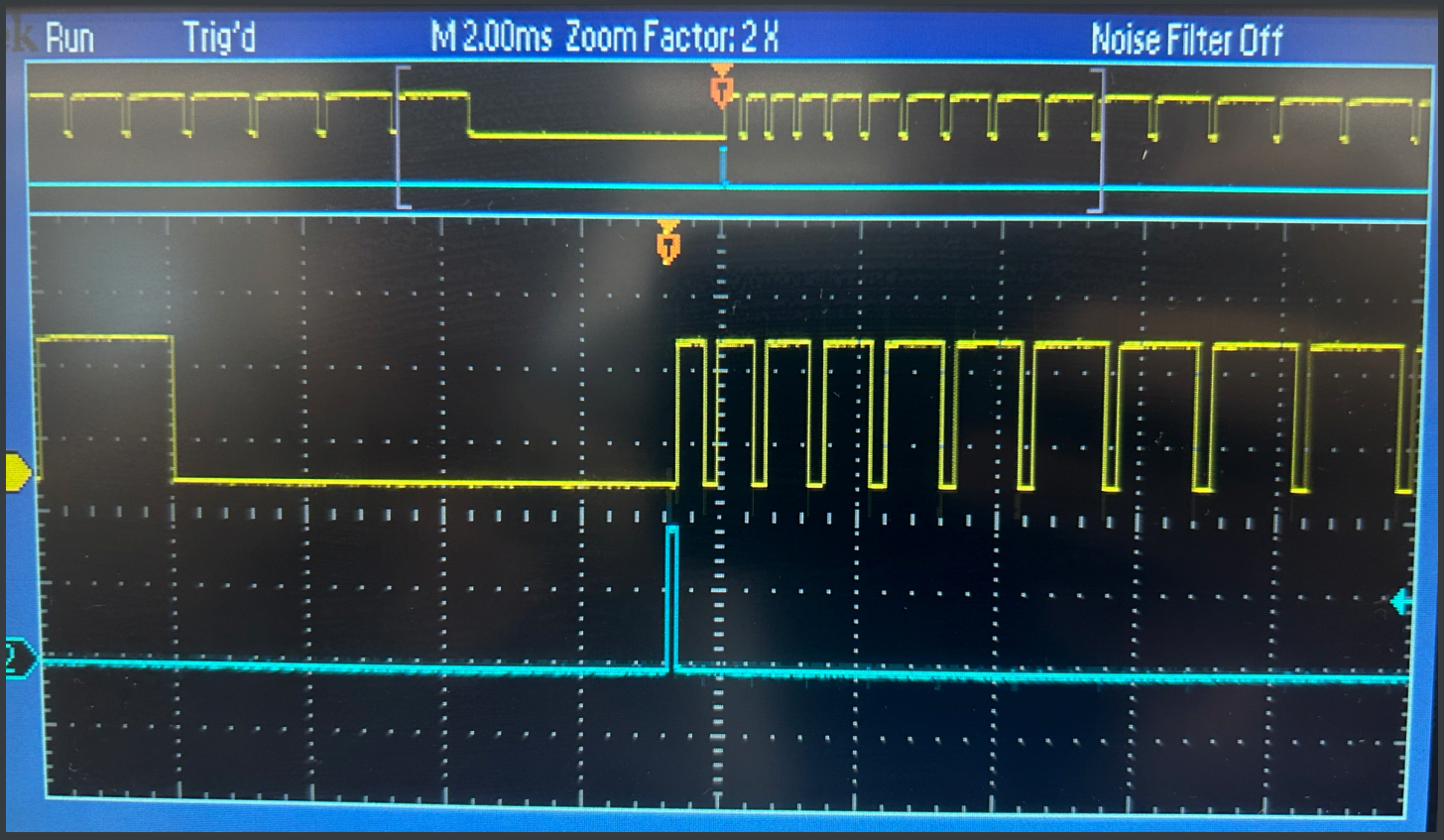
Above is the initial figure of the DATA and SYNC signal

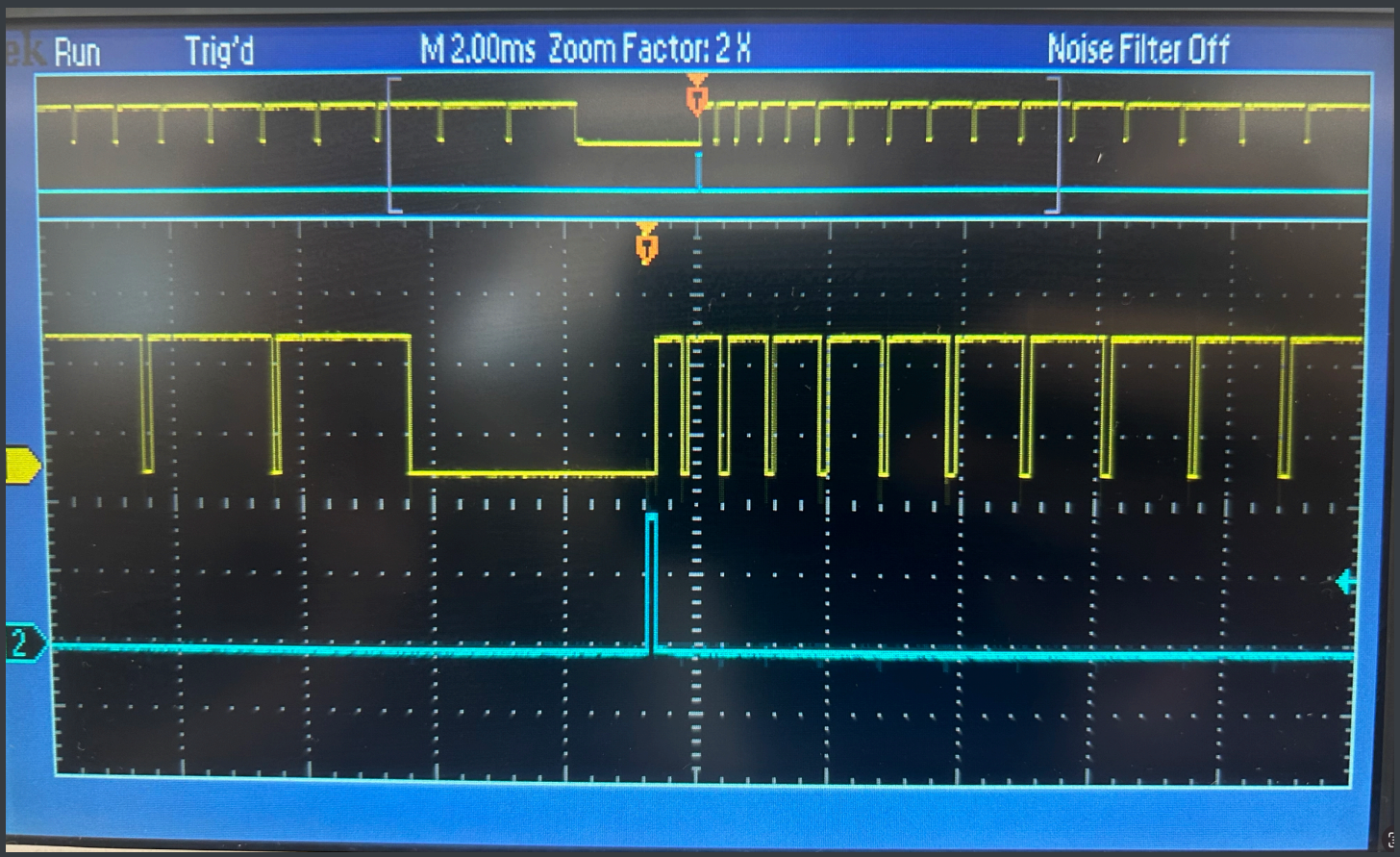
And when we click the button the DATA signal changed into mode4 and the figure turned into



As you can see the intervals of b and d turned half but b = 100us is too small to see the change

We can see comparation in bigger pictures below


Run
Trig'd
M 2.00ms Zoom Factor: 2 X
Noise Filter Off


Run
Trig'd
M 2.00ms Zoom Factor: 2 X
Noise Filter Off

4. Images of Hardware Circuit including ESP32 PCB, Push Buttons and LEDs

Here's the images of hardware circuit in several directions