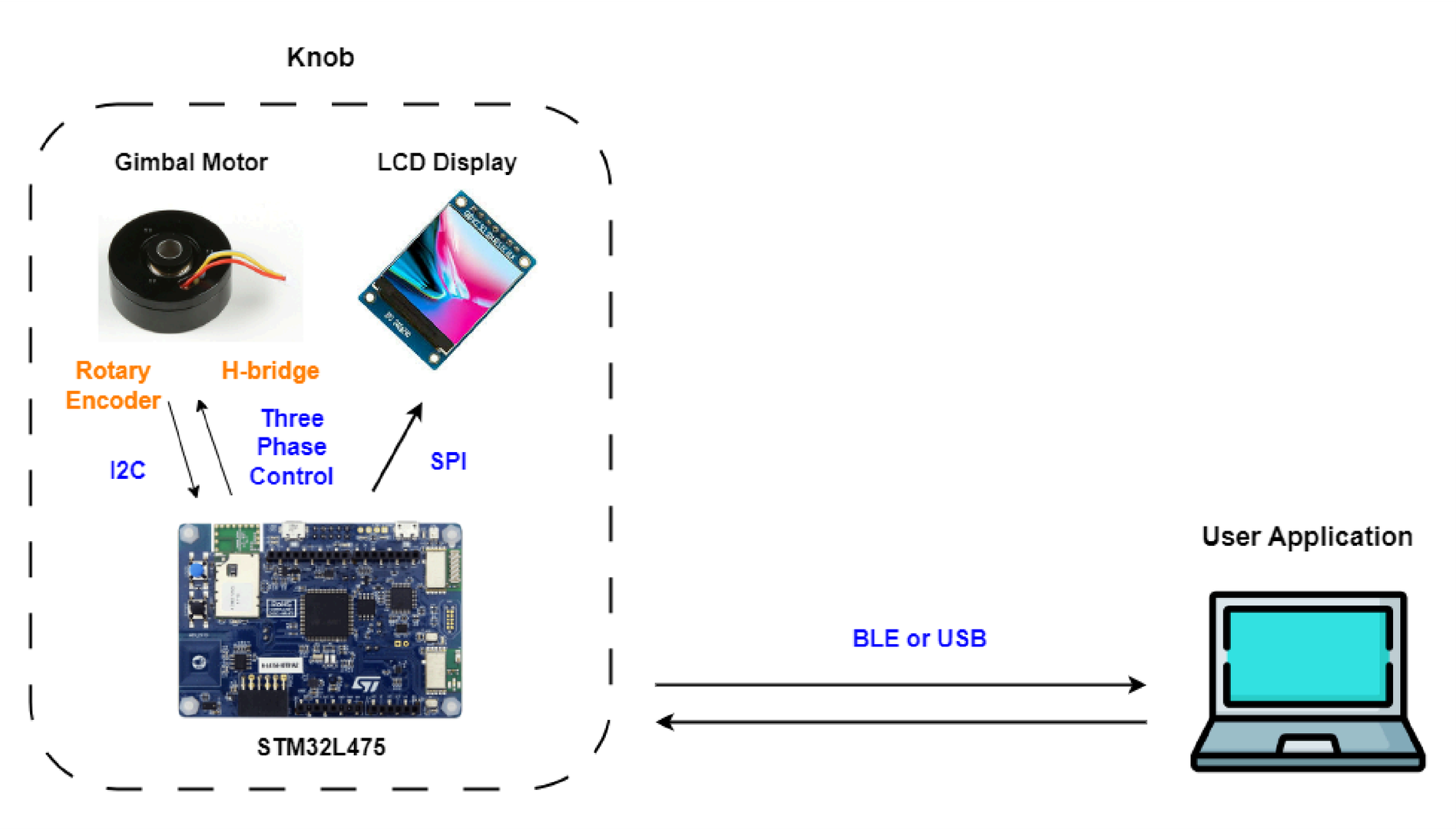
Embedded System Lab
Final Report

Group 8 林萬荃、周君桓、陳柏睿

Computer knob with FOC controlled gimbal motor

System diagram



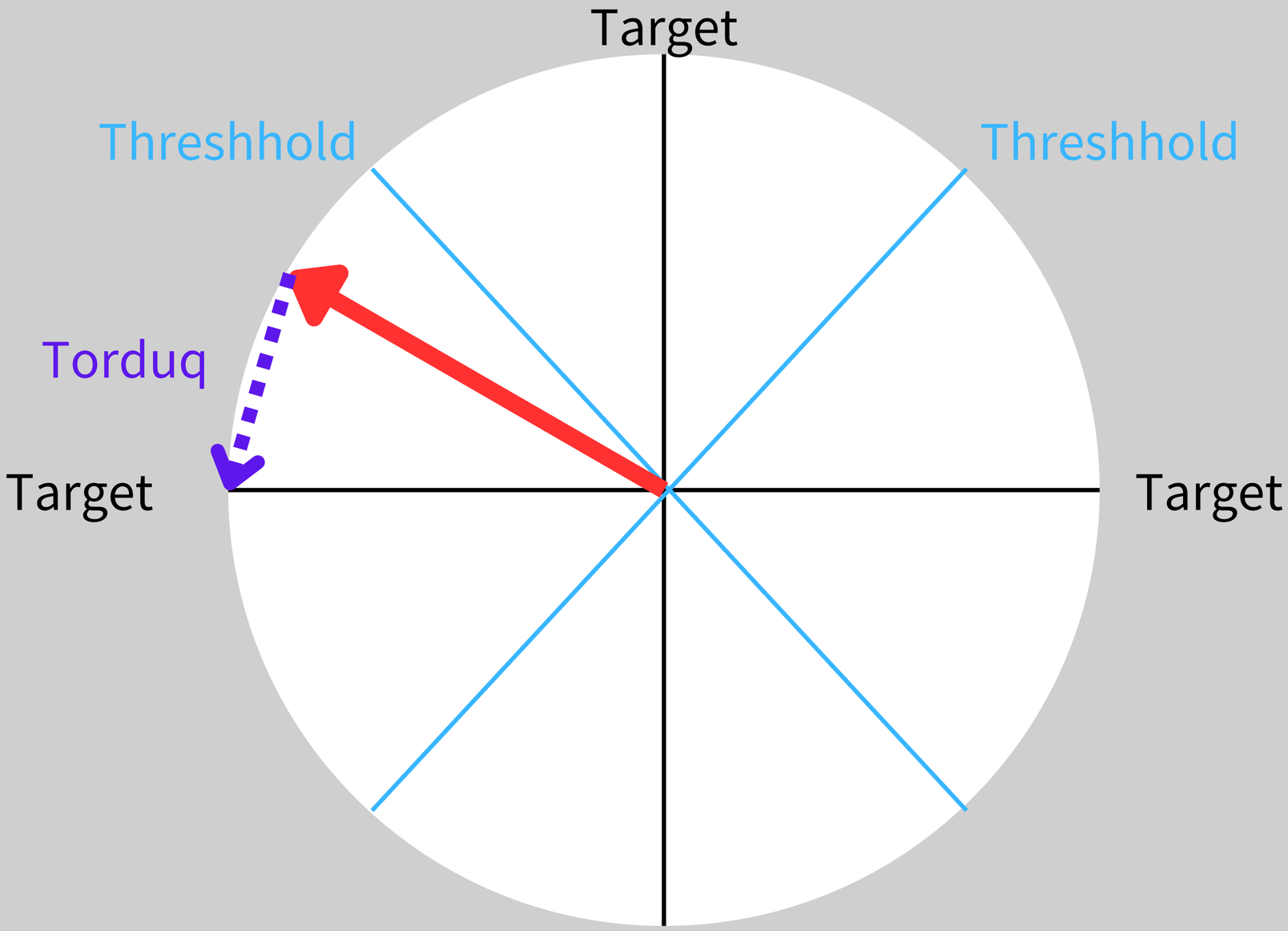
Techniques used

- **Motor**
  - FOC(Field Oriented Control)
  - PID Controller
  - I2C Communication
- **LCD Display**
  - SPI Communication
  - DMA
- **Device Connection**
  - USB HID
  - BLE / GATT
- **Project Stack**
  - STM32 HAL driver
  - CMake & Ninja build

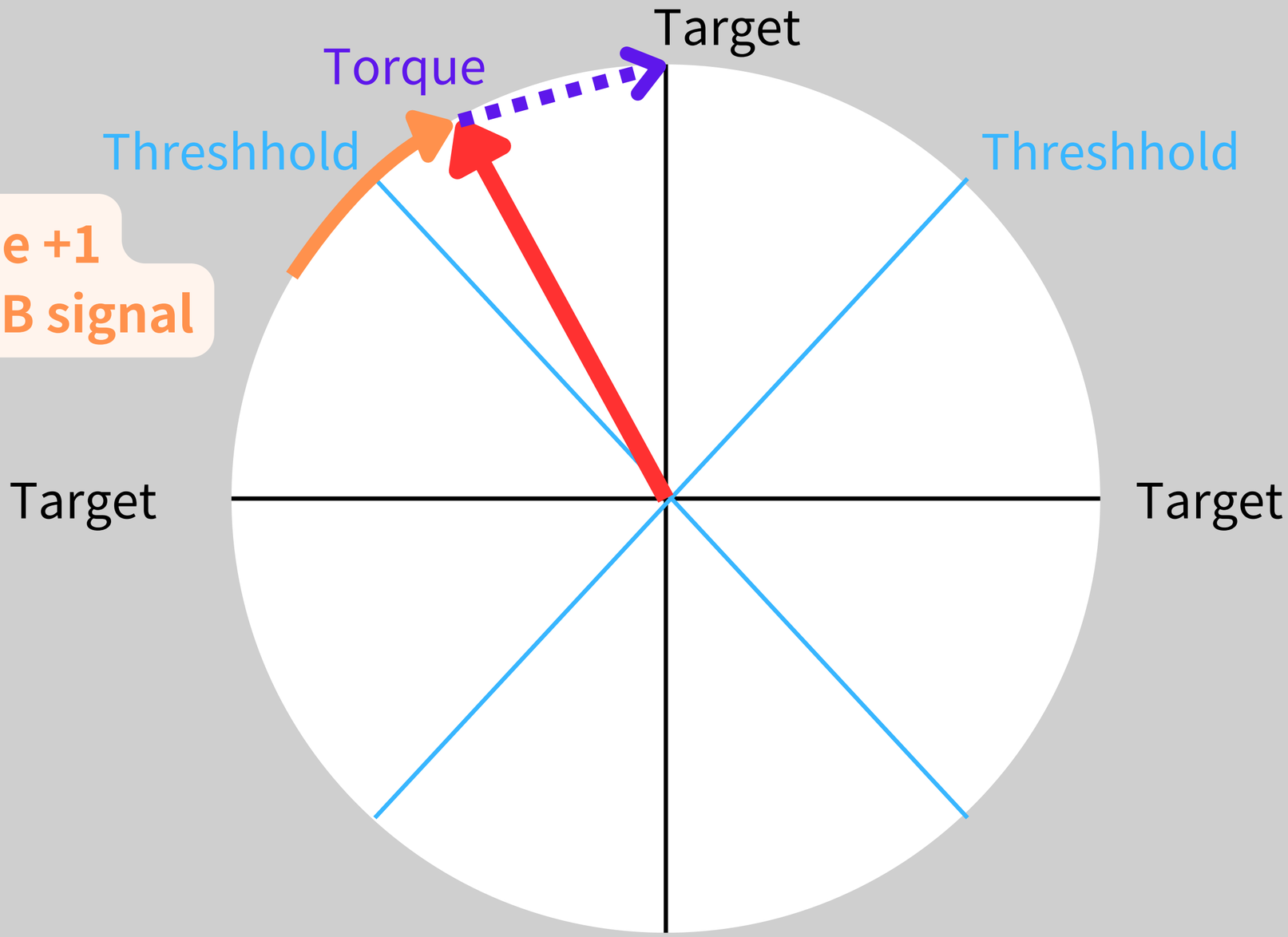
System integration

- FOC motor control
- USB volume control
- TFT LCD control through SPI
- BLE task bridged with Windows application

FOC with USB volume control



FOC with USB volume control



FOC with USB volume control

TFT LCD control - SPI & DMA

Draw volume

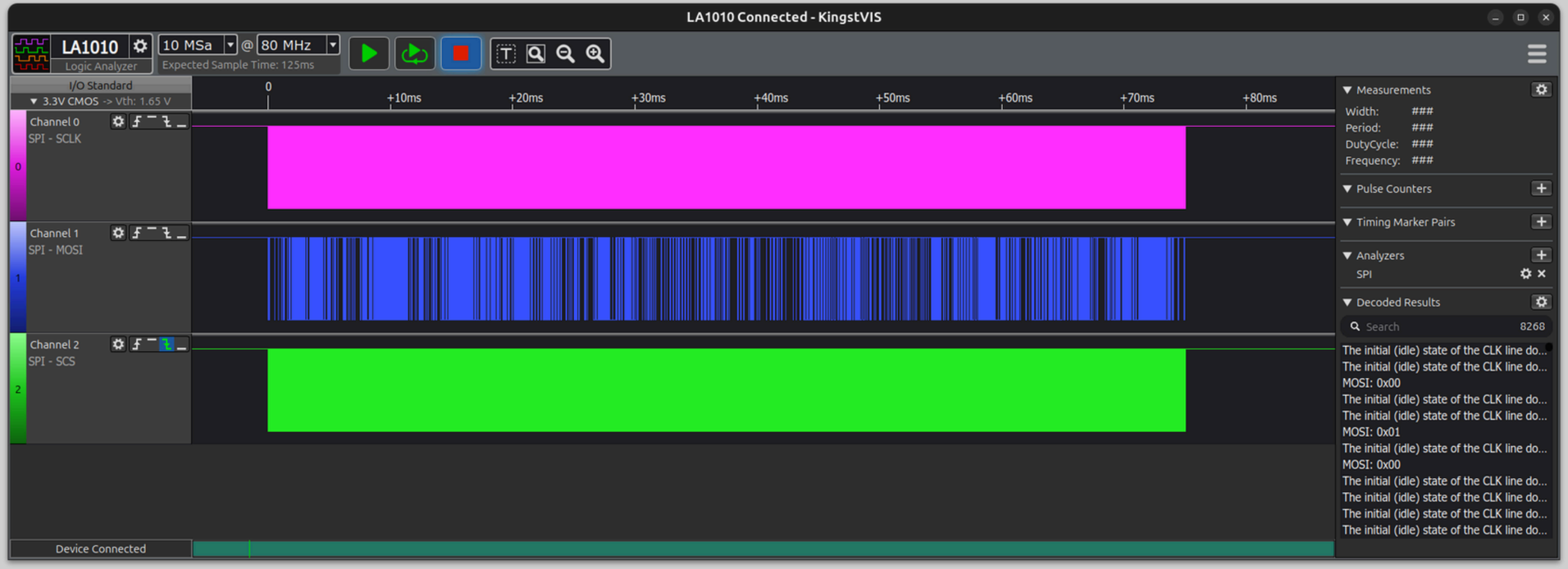
```
if (VOL_FLAG != 0)
    {
      sprintf(msg, "Vol: -", Partition_Inx);
      ST7789_WriteString(10, 100, msg, Font_16x26, WHITE, BLACK);
    }
```

Transfer data through DMA

```
if (DMA_MIN_SIZE <= buff_size)
        {
            HAL_SPI_Transmit_DMA(&ST7789_SPI_PORT, buff, chunk_size);
            while (ST7789_SPI_PORT.hdmatx->State != HAL_DMA_STATE_READY)
                ;
        }
```

SPI frame buffer time diagram

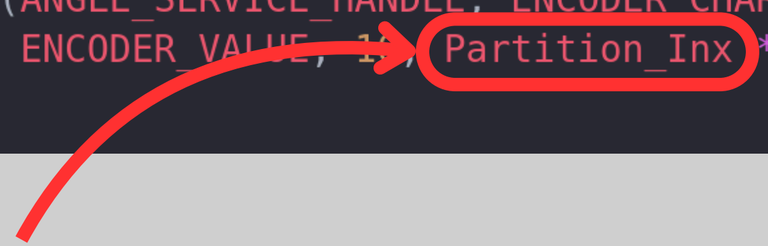
About 70 ms of data transfer handled my DMA



One frame date transfer

BLE integration with main task

```
1  void BLE_Task(void)
2  {
3    if (HAL_GPIO_ReadPin(BLE_INT_GPIO_Port, BLE_INT_Pin))
4    { // if an event occurs let's catch it
5      catchBLE();
6    }
7    updateSignedMillesimal(ANGLE_SERVICE_HANDLE, ENCODER_CHAR_HANDLE,
8                           ENCODER_VALUE, 1, Partition_Inx * 10);
9  }
```



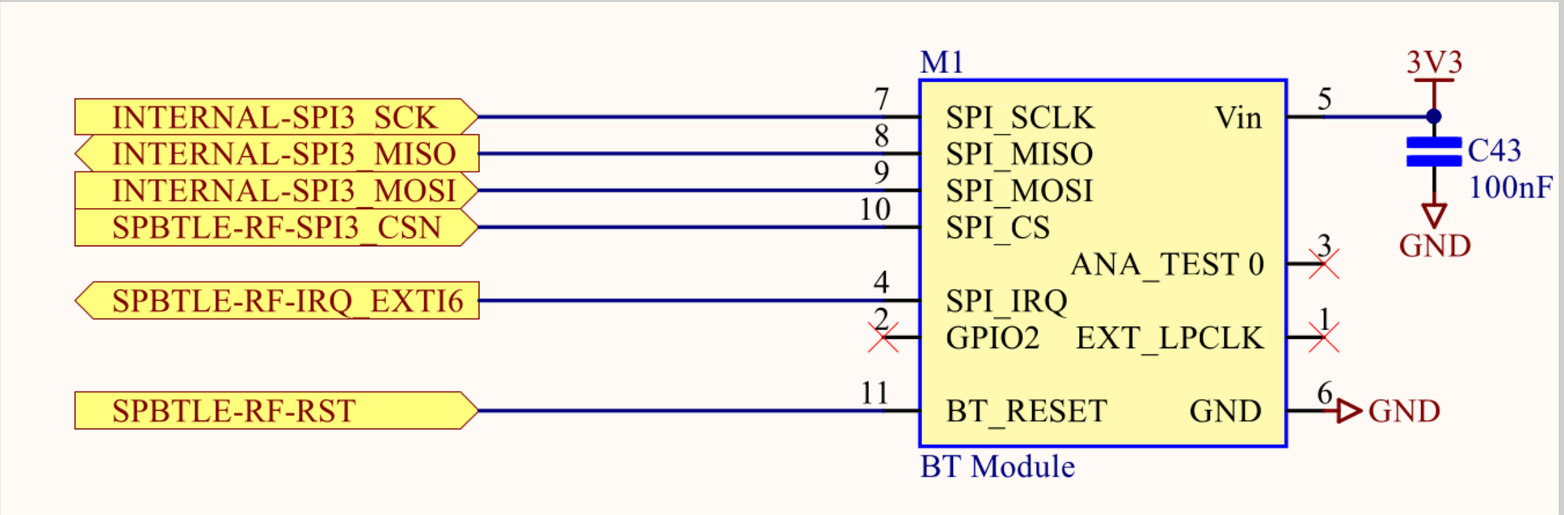
According to the dial angular position (Partition index) update the BLE service value

BLE: HCL & SPI

Using HCI command (legacy) interface with BLE on board module
For example: this command start the server advertisement

```
uint8_t ACI_GAP_SET_DISCOVERABLE[]={0x01,0x83,0xfc,0xff,0x00,0x40,0x06,0x40,0x06,0x01,0x00,0xff,0x09};
```

It is interfaced with SPI



Self-defined BLE Service on STM32

Self defined GATT characteristics

```
24 uint8_t UUID_ANGLE_SERVICE[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x05,0x50,0x05,0x00};
25 uint8_t ANGLE_SERVICE_HANDLE[2];
26
27 // Characteristic
28 uint8_t UUID_CHAR_ENCODER[] = {0x01,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x50,0x05,0x00};
29 uint8_t ENCODER_CHAR_HANDLE[2];
30 uint8_t ENCODER_VALUE[]={'{','\"','A','n','g','l','e','\"',':','\"','+','0','0','0','0','\"','}'};
31
32 uint8_t UUID_CHAR_OUTPUT[] = {0x01,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x50,0x05,0x01};
33 uint8_t OUTPUT_CHAR_HANDLE[2];
34 uint8_t OUTPUT_VALUE[]={'{','\"','A','n','g','l','e','\"',':','\"','+','0','0','0','0','\"','}'};
```

```
// Add Service
addService(UUID_ANGLE_SERVICE, ANGLE_SERVICE_HANDLE, SET_ATTRIBUTES(1+2+3*2));
// Add Characteristic
addCharacteristic(UUID_CHAR_ENCODER, ENCODER_CHAR_HANDLE, ANGLE_SERVICE_HANDLE, SET_CONTENT_LENGTH(17), READABLE|NOTIFIBLE);
updateCharValue(ANGLE_SERVICE_HANDLE, ENCODER_CHAR_HANDLE, 0, SET_CONTENT_LENGTH(17), ENCODER_VALUE);

addCharacteristic(UUID_CHAR_OUTPUT, OUTPUT_CHAR_HANDLE, ANGLE_SERVICE_HANDLE, SET_CONTENT_LENGTH(17), WRITABLE);
updateCharValue(ANGLE_SERVICE_HANDLE, OUTPUT_CHAR_HANDLE, 0, SET_CONTENT_LENGTH(17), OUTPUT_VALUE);
```

```
// update encoder value
updateSignedMillesimal(ANGLE_SERVICE_HANDLE, ENCODER_CHAR_HANDLE, ENCODER_VALUE, 10, encoder_value);
```

Find GATT Characteristic on Win11

Find by UUID
(Note: services and characteristic need to be declared static)

```
private static GattDeviceService? gattService;
2 references
private static GattCharacteristic? gattCharacteristic;
```

```
Guid serviceUuid = new Guid("00055005-0000-0000-0001-000000000000");
Guid characteristicUuid = new Guid("00055000-0000-0000-0003-000000000001");
```

Subscribe to GATT Charateristic

Subscribe and add a handler

```
if (properties.HasFlag(GattCharacteristicProperties.Notify))
{
    GattCommunicationStatus status = await characteristic.WriteClientCharacteristicConfigurationDescriptorAsync(
        GattClientCharacteristicConfigurationDescriptorValue.Notify);
    if (status == GattCommunicationStatus.Success)
    {
        characteristic.ValueChanged += Characteristic_ValueChanged;
    }
}
```

Turn the bytes read into string and extract the values to set the volume of the PC

```
var reader = DataReader.FromBuffer(args.CharacteristicValue);
byte[] input = new byte[reader.UnconsumedBufferLength];
reader.ReadBytes(input);
string str = Encoding.ASCII.GetString(input);
Console.WriteLine($"receive: {str}");
int angleValue = ExtractValue(str);
Console.WriteLine($"Extracted value: {angleValue}");
VolumeController.SetVolume(angleValue);
```

Windows Side Extract Value read

Value format: {” <Angle>” :” +deg”}

```
private static int ExtractValue(string str)
{
    str = str.Trim(new char[] { '{', '}' });
    string[] split_str = str.Split(':');
    str = split_str[1].Trim(new char[] { '"', ' ' });
    str = str.TrimStart('+');
    return int.Parse(str);
}
```

Extract result: Angle

Set Volume with Windows API

Using CoreAudio API

```
1   using AudioSwitcher.AudioApi.CoreAudio;
2
3   namespace GATT_Client_Win11;
4
    You, 2 days ago | 1 author (You) | 0 references
5   class VolumeController
6   {
        1 reference
7       private static CoreAudioController audioController = new CoreAudioController();
        1 reference
8       private static CoreAudioDevice playbackDevice = audioController.DefaultPlaybackDevice;
9
        0 references
10      public static void SetVolume(int value)
11      {
12          playbackDevice.Volume = value;
13      }
14  }       You, 2 days ago • move windows GATT
```

Monitoring CPU Usage (WIP)

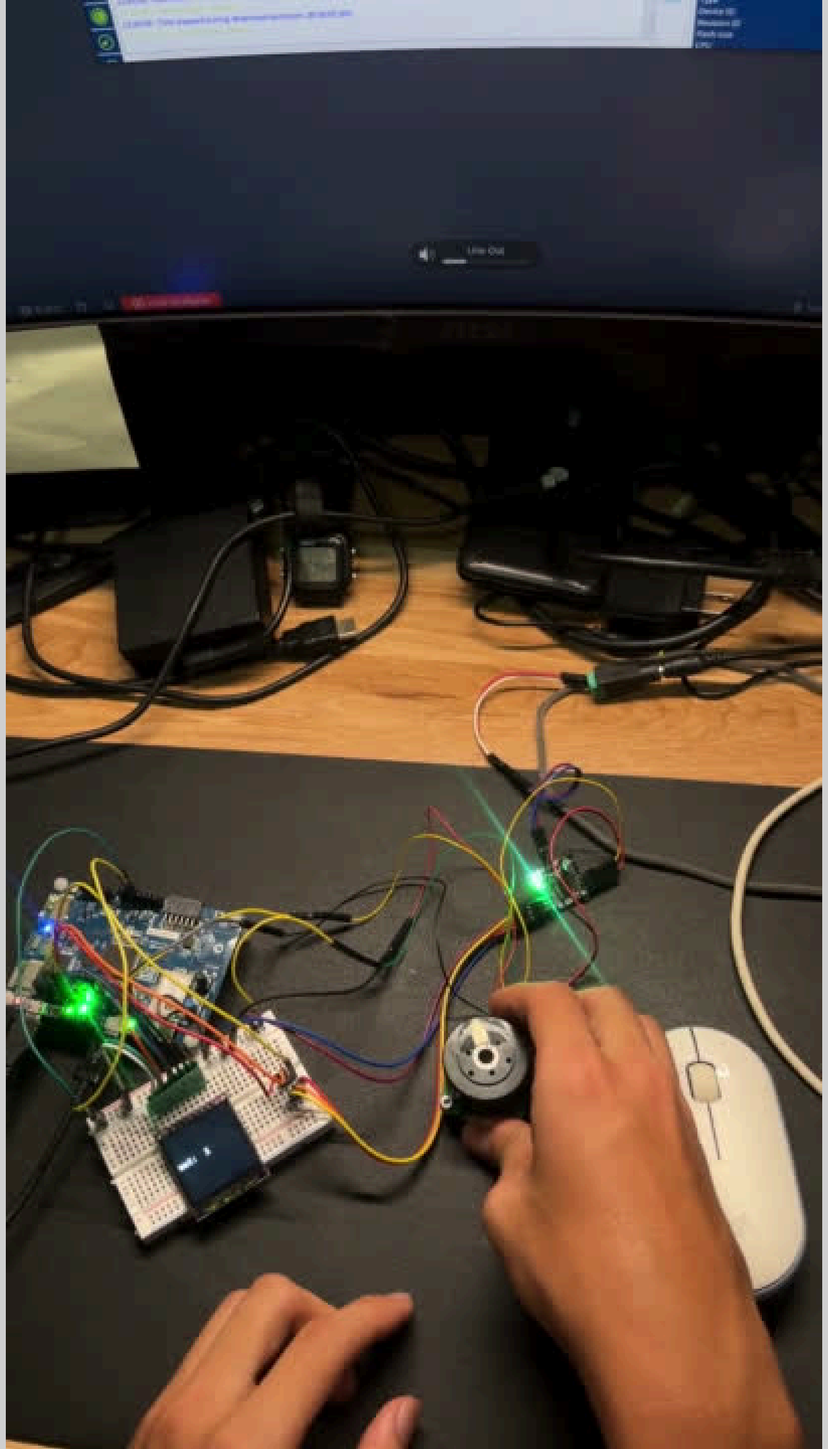
Using performance counter in System.Diagnostics API

```
PerformanceCounter cpuCounter = new PerformanceCounter("Processor", "% Processor Time", "_Total");
```

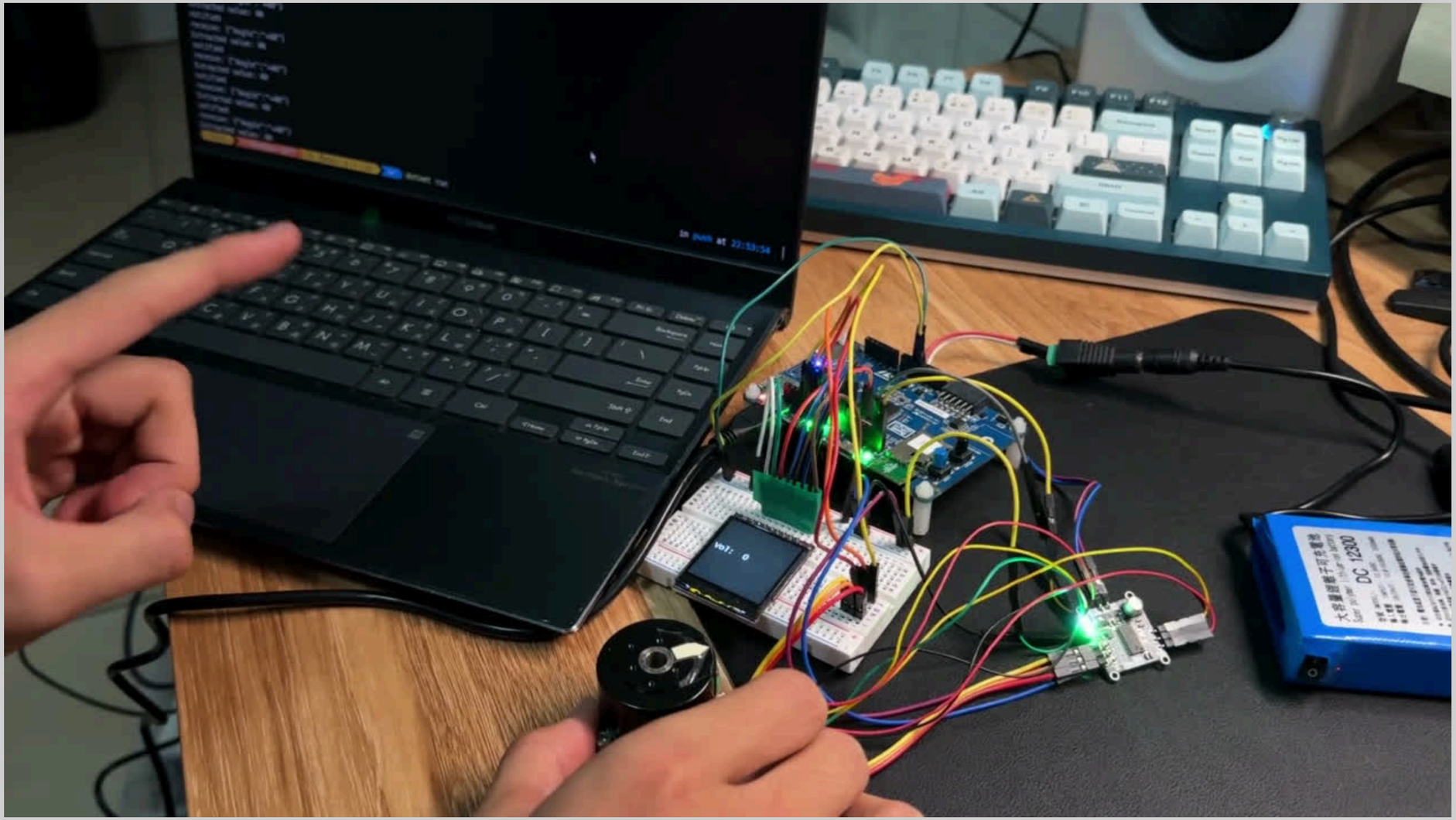
Calculate avverage CPU usage

```
while (true)
{
    float[] samples = new float[5];
    for (int i = 0; i < samples.Length; i++)
    {
        samples[i] = cpuCounter.NextValue();
        Thread.Sleep(200); // Sample every 200ms for a total of 1 second
    }

    float averageCpuUsage = samples.Average();

    // Print the average CPU usage percentage
    Console.WriteLine("Average CPU Usage: {0}%", averageCpuUsage);

    // Wait for a second before getting the next set of samples
    Thread.Sleep(100);
}
```

Demo (USB)



Demo(BLE)



Reference

- https://github.com/Floyd-Fish/ST7789-STM32
- https://github.com/iot2tangle/STM32_B-L475E-IOT01A/tree/main/BLE-sender