Flotylla Sokólska	**Protokół projektu układu testującego lokalizację trilateracją w systemie Człowiek za burtą przy użyciu BLE**		Data wystawienia: 21/07/2021	
			Doc#	3/CZB/002
	Nr wniosku NCBR:	POIR.01.01.01-00-0196/19	Nazwa projektu:	Smart Yacht
	Rozpoczęcie testów:	14-07-2021	Zakończenie testów:	21-07-2021

1. Założenia

Celem jest zbudowanie układu testującego metodę trilateracji. Odbiornik musi umożliwiać odczyt mocy sygnału z dwóch nadajników iBeacon z dystansu <5m i pokazywać ją na wyświetlaczu, wraz z fragmentem adresu MAC. Powinien też być mobilny – wymagany jest odczyt mocy sygnału z różnych miejsc(symulacja wielu odbiorników).

Nadajnik iBeacon musi charakteryzować się małymi wymiarami, które umożliwią wbudowanie go w opaskę na nadgarstek na dalszych etapach projektu.

2. Hardware

2.1 Skaner

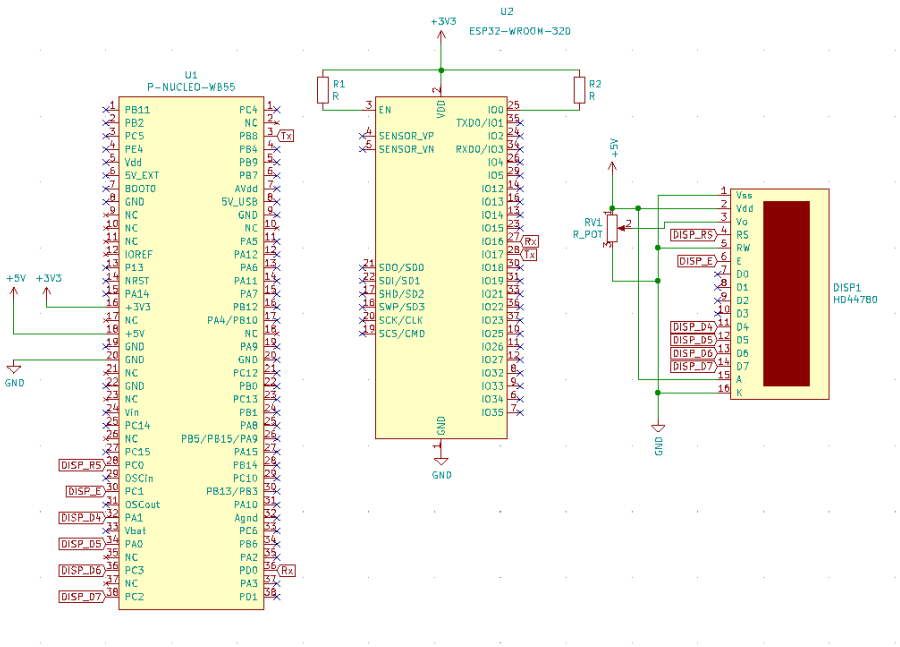

Figura 1: Schemat testowego skanera

Bazą dla układu testującego jest płytka P-NUCLEO-WB55, do której przy pomocy interfejsu UART został podłączony SoC ESP32-WROOM-32D który posłuży jako skaner Bluetooth Low Energy(BLE). Interfejs użytkownika stanowi wyświetlacz alfanumeryczny LCD. Całość została zasilona przez USB przy pomocy power banka- układ skanera będzie przenoszony pomiędzy punktami pomiarowymi.

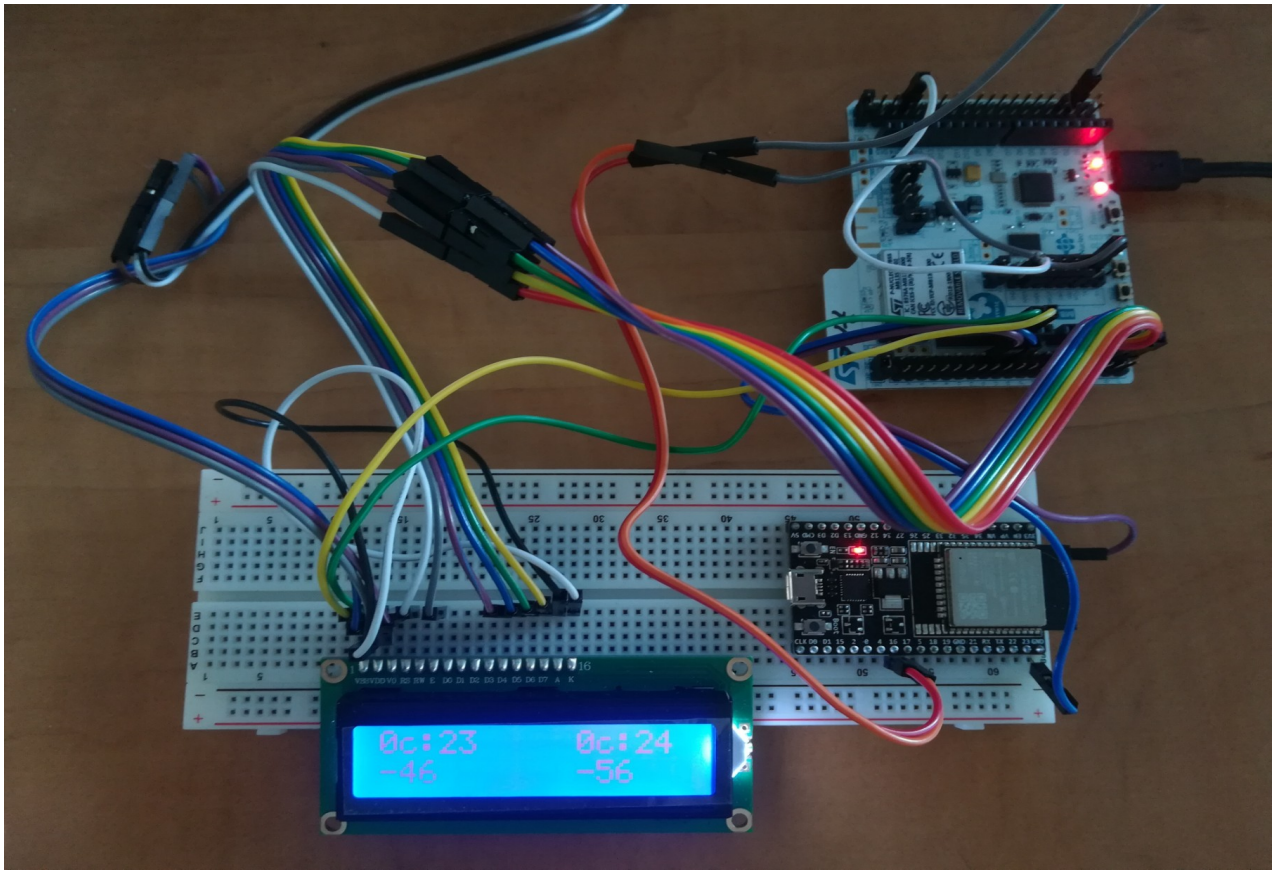

Figura 2: Testowy układ skanera

2.2 Transmiter

Jako urządzenie lokalizowane posłużył moduł iNode Beacon- zasilany bateryjnie konfigurowalny nadajnik Bluetooth Low Energy, zwany dalej „Tagiem”.
Służy on do przesyłania ramek rozgłoszeniowych, tzw. Advertisements zgodnych ze standardem BLE zawierających UUID oraz adres MAC taga.

Dla projektu istotna jest modyfikacja standardowego UUID służącego do lokalizowania obiektu- nie można dopuścić do tego, aby został on pomylony ze standardowym tagiem, który ktoś może mieć przy sobie. Oprócz tego iNode Beacon pozwala również na wybór pożądanej mocy nadawania spośród poniższych wartości:

5%	10%	12,5%	25%	33%	50%	80%	100%

-18dBm	-14dBm	-10dBm	-6dBm	-2dBm	+2dBm	+5dBm	+8dBm

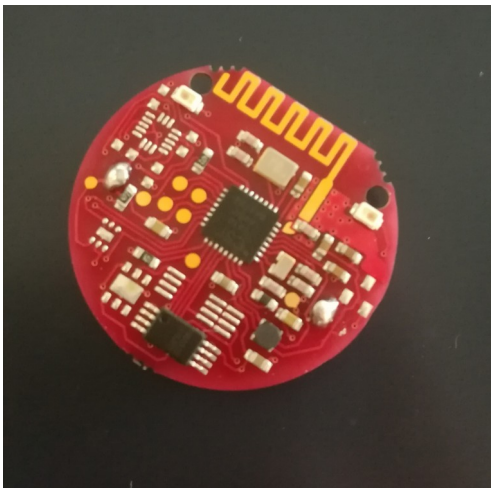

Figura 3: Tag bez obudowy

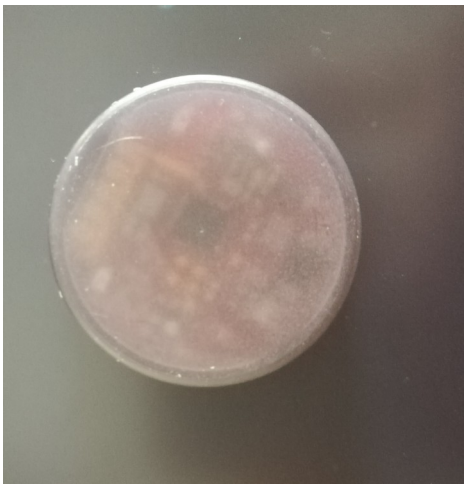

Figura 4: Tag w obudowie

3. Software

3.1 Moduł odbiorczy

ESP32 został wyposażony w standardowy AT Firmware w wersji 2.1.0.0. Jest to oprogramowanie producenta(Espressif) pozwalające na komunikację z ESP32 przy pomocy interfejsu UART standardem Hayes AT, powszechnie używanym do komunikacji komputerów z modemami.

3.2 iBeacon

jest fabrycznie wyposażony w firmware rozsyłający ramki rozgłoszeniowe(tzw. Advertisements) zawierające prefix, UUID oraz adres MAC, w standardzie BLE(Bluetooth Low Energy). W trybie normalnej pracy nie jest możliwe nawiązanie połączenia z modułem. Po wyjęciu i włożeniu baterii iBeacon przechodzi w tryb programowania na 2 minuty. W tym czasie, łącząc się z nim przy pomocy komputera lub telefonu używając dedykowanej aplikacji iNode Setup można zmodyfikować moc nadawania, częstotliwość wysyłania ramek a także zmienić UUID. Do testów wybrana została moc nadawania 80%. Zmieniony został również prefix ramki iBeacon w celu wykluczenia pomyłek.

Standardowa ramka iBeacon	
Prefix:	02 01 06 1A FF 4C 00 **02 15**
UUID:	FDA50693-A4E2-4FB1-AFCF-C6EB07647825
ID sklepu:	27 B7
ID lokalizacji:	F2 06
Moc nadawania:	C5

Modyfikacja ramki	
Prefix:	02 01 06 1A FF 4C 00 **42 38**
UUID:	FDA50693-A4E2-4FB1-AFCF-C6EB07647825
ID sklepu:	27 B7
ID lokalizacji:	F2 06
Moc nadawania:	C5

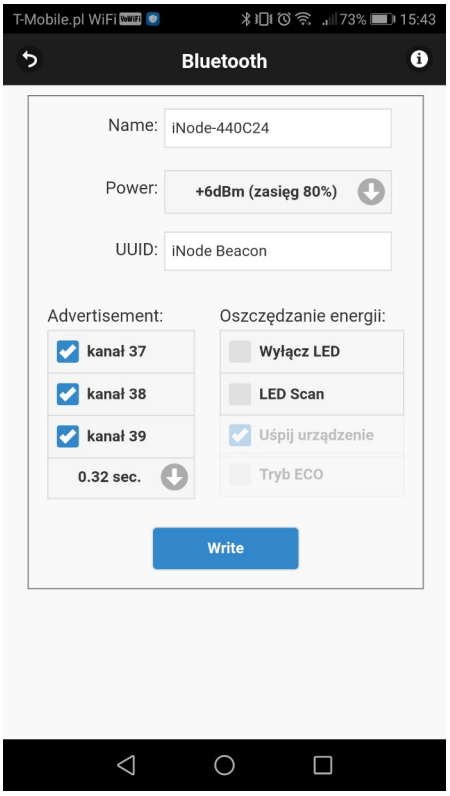

Figura 5: iNode Setup- moc nadawania

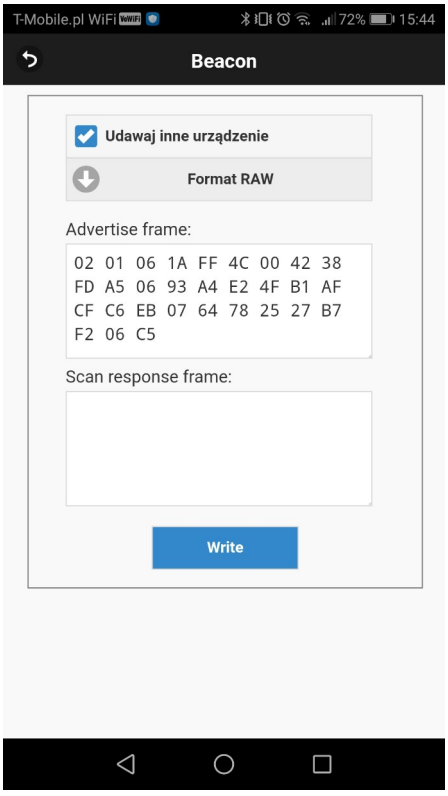

Figura 6: iNode Setup- modyfikacja ramki rozgłoszeniowej

3.3 Skaner

STM32 został wyposażony w program testowy mający za zadanie uruchomienie ESP32, skonfiguranie go jako klienta BLE oraz uruchomienie skanowania. Następnie STM odbiera od ESP32 ramki rozgłoszeniowe wszystkich urządzeń BLE w zasięgu, szuka pośród nich zmodyfikowanych ramek iBeacon a następnie wypisuje na wyświetlaczu fragment adresu MAC dwóch pierwszych znalezionych urządzeń wraz z ich aktualnym zasięgiem, co pozwala na późniejsze obliczenie dystansu.

Zostały oprogramowane również dwie diody. Zielona zapala się kiedy zostaną odebrane dane od ESP32, natomiast niebieska kiedy w odebranej ramce znajdzie się iBeacon.

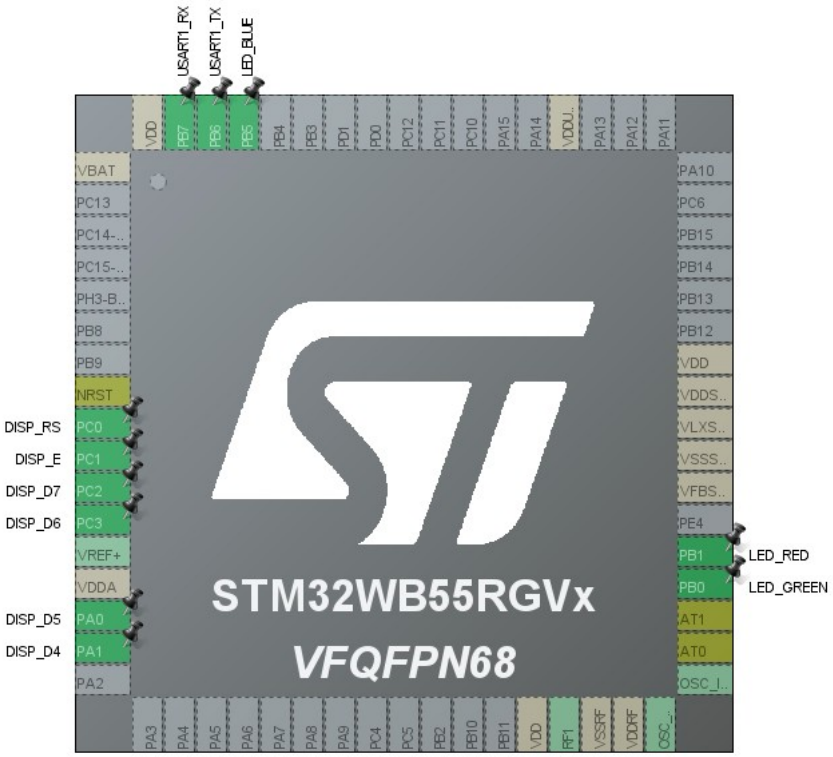

Figura 7: Pinout STM32 w układzie skanera

Kod programu, plik main.c:

```
#include "main.h”
#include "../Display/an_disp.h"
#include <string.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <stdbool.h>

#include "../Interrupts/interrupts.h"
#include "../AT/AT.h"
IWDG_HandleTypeDef hiwdg;

UART_HandleTypeDef huart1;
DMA_HandleTypeDef hdma_usart1_rx;

uint8_t receiveBuffer[BUFFER_SIZE];
char receiveData[BUFFER_SIZE];

char iBeacon[61];

char signal[3];
char MAC[17];
int8_t signalInt;

char *pointerToUUID;
char *pointerToSignal;
char *pointerToMAC;

struct beacon {
	char MAC[18];
	char signal[4];
	bool busy;
	int8_t filteredSignal;
} beacon1, beacon2, beacon3, beaconBusyPattern;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_USART1_UART_Init(void);
```

```
static void MX_IWDG_Init(void);
static void MX_NVIC_Init(void);



int main(void) {

	HAL_Init();
	SystemClock_Config();
	MX_GPIO_Init();
	MX_DMA_Init();
	MX_USART1_UART_Init();
	MX_IWDG_Init();
	MX_NVIC_Init();
	memset(&beacon1, '0', sizeof(beacon1));
	memset(&beacon2, '0', sizeof(beacon2));
	memset(&beacon3, '0', sizeof(beacon3));

	beacon1.filteredSignal = -30;
	beacon2.filteredSignal = -30;
	beacon3.filteredSignal = -30;



	beacon1.busy = false;
	beacon2.busy = false;
	beacon3.busy = false;



	strcpy(beaconBusyPattern.MAC, "00:00:00:00:00:00");
	strcpy(beaconBusyPattern.signal, "-99");
	beaconBusyPattern.busy = true;

	lcdInit();
	lcdStr("test");
```

```
	HAL_UARTEx_ReceiveToIdle_DMA(AT_UART, receiveBuffer, BUFFER_SIZE);

	HAL_Delay(200);
	ATrfPower(78, 0, 0, 0);
	ATbleInit(AT_BLE_CLIENT);
	ATbleScanEnable(0);

	strcpy(iBeacon,
			"0201061aff4c004238fda50693a4e24fb1afcfc6eb0764782527b7f206c5");

	flagReceived = 0;

	HAL_IWDG_Init(&hiwdg);
	while (1) {

		if (flagReceived == 1) {

			HAL_IWDG_Refresh(&hiwdg);
			memcpy(&receiveData, &receiveBuffer, BUFFER_SIZE);

			//search for ibeacon UUID in received data.
			pointerToUUID = strstr(receiveData, iBeacon);
			pointerToSignal = pointerToUUID - 4;
			pointerToMAC = pointerToUUID - 23;

			if (pointerToUUID != 0) { //if beacon detected

				do {

					HAL_GPIO_WritePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin,
1);

					//read signal
```

```
                        memcpy(signal, pointerToSignal, 3);

                        //read MAC
                        memcpy(MAC, pointerToMAC, 17);

                        if (signal[0] == '-' && MAC[2] == ':') { //check
values

                              if (memcmp(MAC, beacon1.MAC, 17) == 0) {
                                    memcpy(beacon1.signal, signal, 3);
                                    beacon1.busy = true;

                                    if (atoi(beacon1.signal) / 10
                                                > beacon1.filteredSignal) {
                                          beacon1.filteredSignal++;
                                    } else if (atoi(beacon1.signal) / 10
                                                < beacon1.filteredSignal) {
                                          beacon1.filteredSignal--;
                                    }

                              } else if (memcmp(MAC, beacon2.MAC, 17) == 0) {
                                    memcpy(beacon2.signal, signal, 3);
                                    beacon2.busy = true;

                                    if (atoi(beacon2.signal) / 10
                                                > beacon2.filteredSignal) {
                                          beacon2.filteredSignal++;
                                    } else if (atoi(beacon2.signal) / 10
                                                < beacon2.filteredSignal) {
                                          beacon2.filteredSignal--;
                                    }

                              } else if (memcmp(MAC, beacon3.MAC, 17) == 0) {
                                    memcpy(beacon3.signal, signal, 3);
```

```
        beacon3.busy = true;

        if (atoi(beacon3.signal) / 10
                    > beacon3.filteredSignal) {
            beacon3.filteredSignal++;
        } else if (atoi(beacon3.signal) / 10
                    < beacon3.filteredSignal) {
            beacon3.filteredSignal--;
        }

    } else {

        if (beacon1.busy == false) {
            memcpy(beacon1.MAC, MAC, 17);
            memcpy(beacon1.signal, signal, 3);
            beacon1.busy = true;

        } else if (beacon2.busy == false) {
            memcpy(beacon2.MAC, MAC, 17);
            memcpy(beacon2.signal, signal, 3);
            beacon2.busy = true;
        } else if (beacon3.busy == false) {
            memcpy(beacon3.MAC, MAC, 17);
            memcpy(beacon3.signal, signal, 3);
            beacon3.busy = true;
        }

    }

    // print MAC/signal on display

    lcdLocate(0, 0);
    lcdChar(beacon1.MAC[12]);
    lcdChar(beacon1.MAC[13]);
```

```
                        lcdChar(beacon1.MAC[14]);
                        lcdChar(beacon1.MAC[15]);
                        lcdChar(beacon1.MAC[16]);
                        lcdLocate(0, 1);
                        lcdInt(beacon1.filteredSignal);
                        lcdStr("    ");

                        lcdLocate(10, 0);
                        lcdChar(beacon2.MAC[12]);
                        lcdChar(beacon2.MAC[13]);
                        lcdChar(beacon2.MAC[14]);
                        lcdChar(beacon2.MAC[15]);
                        lcdChar(beacon2.MAC[16]);
                        lcdLocate(10, 1);
                        lcdInt(beacon2.filteredSignal);
                        lcdStr("   ");


                    }

                    //clear needle
                    memset(pointerToUUID, '0', 59);

                    //search for next needle
                    pointerToUUID = strstr(receiveData, iBeacon);
                    pointerToSignal = pointerToUUID - 4;
                    pointerToMAC = pointerToUUID - 23;

                    HAL_GPIO_WritePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin,
0);

                } while (pointerToUUID != 0);

            }
```

```
                flagReceived = 0;
                HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 0);

void SystemClock_Config(void) {
	RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
	RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };
	RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = { 0 };

	__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
	RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI
			| RCC_OSCILLATORTYPE_LSI1 | RCC_OSCILLATORTYPE_MSI;
	RCC_OscInitStruct.HSIState = RCC_HSI_ON;
	RCC_OscInitStruct.MSIState = RCC_MSI_ON;
	RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
	RCC_OscInitStruct.MSICalibrationValue = RCC_MSICALIBRATION_DEFAULT;
	RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
	RCC_OscInitStruct.LSIState = RCC_LSI_ON;
	RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
	RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
	RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV1;
	RCC_OscInitStruct.PLL.PLLN = 32;
	RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
	RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
	RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
	if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
		Error_Handler();

	RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK4 | RCC_CLOCKTYPE_HCLK2
			| RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_PCLK1
			| RCC_CLOCKTYPE_PCLK2;
	RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
	RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
	RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
```

```
	RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
	RCC_ClkInitStruct.AHBCLK2Divider = RCC_SYSCLK_DIV2;
	RCC_ClkInitStruct.AHBCLK4Divider = RCC_SYSCLK_DIV1;

	if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK) {
		Error_Handler();
	}
	PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_SMPS
			| RCC_PERIPHCLK_USART1;
	PeriphClkInitStruct.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
	PeriphClkInitStruct.SmpsClockSelection = RCC_SMPSCLKSOURCE_HSI;
	PeriphClkInitStruct.SmpsDivSelection = RCC_SMPSCLKDIV_RANGE1;
	if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK) {
		Error_Handler();
	}
static void MX_NVIC_Init(void) {
	/* USART1_IRQn interrupt configuration */
	HAL_NVIC_SetPriority(USART1_IRQn, 0, 0);
	HAL_NVIC_EnableIRQ(USART1_IRQn);
	/* DMA1_Channel1_IRQn interrupt configuration */
	HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
	HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);
}
static void MX_IWDG_Init(void) {

	hiwdg.Instance = IWDG;
	hiwdg.Init.Prescaler = IWDG_PRESCALER_32;
	hiwdg.Init.Window = 4095;
	hiwdg.Init.Reload = 4095;
	if (HAL_IWDG_Init(&hiwdg) != HAL_OK) {
		Error_Handler();
	}
static void MX_USART1_UART_Init(void) {
```

```
	huart1.Instance = USART1;
	huart1.Init.BaudRate = 115200;
	huart1.Init.WordLength = UART_WORDLENGTH_8B;
	huart1.Init.StopBits = UART_STOPBITS_1;
	huart1.Init.Parity = UART_PARITY_NONE;
	huart1.Init.Mode = UART_MODE_TX_RX;
	huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
	huart1.Init.OverSampling = UART_OVERSAMPLING_16;
	huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
	huart1.Init.ClockPrescaler = UART_PRESCALER_DIV1;
	huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
	if (HAL_UART_Init(&huart1) != HAL_OK) {
		Error_Handler();
	}
	if (HAL_UARTEx_SetTxFifoThreshold(&huart1, UART_TXFIFO_THRESHOLD_1_8)
			!= HAL_OK) {
		Error_Handler();
	}
	if (HAL_UARTEx_SetRxFifoThreshold(&huart1, UART_RXFIFO_THRESHOLD_1_8)
			!= HAL_OK) {
		Error_Handler();
	}
	if (HAL_UARTEx_DisableFifoMode(&huart1) != HAL_OK) {
		Error_Handler();
	}
static void MX_DMA_Init(void) {

	__HAL_RCC_DMAMUX1_CLK_ENABLE();
	__HAL_RCC_DMA1_CLK_ENABLE();

static void MX_GPIO_Init(void) {
	GPIO_InitTypeDef GPIO_InitStruct = { 0 };
	__HAL_RCC_GPIOC_CLK_ENABLE();
	__HAL_RCC_GPIOA_CLK_ENABLE();
```

```
	__HAL_RCC_GPIOB_CLK_ENABLE();
	HAL_GPIO_WritePin(GPIOC,
	DISP_RS_Pin | DISP_E_Pin | DISP_D7_Pin | DISP_D6_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, DISP_D5_Pin | DISP_D4_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, LED_GREEN_Pin | LED_RED_Pin | LED_BLUE_Pin,
			GPIO_PIN_RESET);

	GPIO_InitStruct.Pin = DISP_RS_Pin | DISP_E_Pin | DISP_D7_Pin | DISP_D6_Pin;
	GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
	GPIO_InitStruct.Pull = GPIO_NOPULL;
	GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
	HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
	GPIO_InitStruct.Pin = DISP_D5_Pin | DISP_D4_Pin;
	GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
	GPIO_InitStruct.Pull = GPIO_NOPULL;
	GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
	HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
	GPIO_InitStruct.Pin = LED_GREEN_Pin | LED_RED_Pin | LED_BLUE_Pin;
	GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
	GPIO_InitStruct.Pull = GPIO_NOPULL;
	GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
	HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
void Error_Handler(void) {
	__disable_irq();
	while (1) {
}

#ifdef  USE_FULL_ASSERT
void assert_failed(uint8_t *file, uint32_t line)
{
}
#endif
```

Kod programu, plik AT.c(komunikacja z ESP32):

```
#include "../AT/AT.h"

void ATbleInit(uint8_t role) {

	char command[14]; //for sprintf
	uint8_t commandUint[14]; //for HAL_UART_Transmit- casted

	sprintf(command, "AT+BLEINIT=%d", role); //make AT command

	for (uint8_t i = 0; i <= 11; i++) {
		commandUint[i] = command[i]; //cast command for HAL
	}
	commandUint[12] = '\r';
	commandUint[13] = '\n'; //add line feed

//	commandUint[3] = 'k'; //corrupted message- just for debug.

	HAL_UART_Transmit(AT_UART, commandUint, sizeof(commandUint), AT_TIMEOUT);
	HAL_Delay(AT_INTERVAL);

}

void ATbleScanEnable(uint8_t interval) {
	char command[16];
	uint8_t commandUint[16];

	sprintf(command, "AT+BLESCAN=1,%d", interval);

	for (uint8_t i = 0; i <= 15; i++) {
		commandUint[i] = command[i]; //cast command for HAL
	}
```

```
	commandUint[14] = '\r';
	commandUint[15] = '\n'; //add line feed

	HAL_UART_Transmit(AT_UART, commandUint, sizeof(commandUint), AT_TIMEOUT);
	HAL_Delay(AT_INTERVAL);

}

void ATbleAdvData(char *advData, uint8_t size){

	char command[18+size];
	uint8_t commandUint[18+size];
	char string[size];

	memcpy(string, advData, size);

	sprintf(command, "AT+BLEADVDATA=\"%s\"", string);

	for(uint8_t i = 0; i<=size+18; i++){
		commandUint[i] = command[i];
	}

	commandUint[size+15] = '"';
	commandUint[size+16] = '\r';
	commandUint[size+17] = '\n';

	HAL_UART_Transmit(AT_UART, commandUint, sizeof(commandUint), AT_TIMEOUT);
	HAL_Delay(AT_INTERVAL);

}

void ATbleAdvStart(void){
	uint8_t command[16] = "AT+BLEADVSTART\r\n";
```

```
	HAL_UART_Transmit(AT_UART, command, sizeof(command), AT_TIMEOUT);

	HAL_Delay(AT_INTERVAL);
}

void ATrfPower(uint8_t wifi, uint8_t bleAdv, uint8_t bleScan, uint8_t bleCon){

	union char2int{
		char character[21];
		uint8_t uint[21];
	}txBuffer;

	sprintf(txBuffer.character, "AT+RFPOWER=%d,%d,%d,%d\r\n", wifi, bleAdv,
bleScan, bleCon);

	HAL_UART_Transmit(AT_UART, txBuffer.uint, sizeof(txBuffer.uint), AT_TIMEOUT);
	HAL_Delay(AT_INTERVAL);

	}
```

Kod programu, plik an_disp.c(obsługa wyświetlacza):

```
#include "../Display/an_disp.h"

void lcdSendHalf(uint8_t data) {
	LCD_E_HIGH;
	HAL_GPIO_WritePin(LCD_D4_PORT, LCD_D4_PIN, (data & 0x01));
	HAL_GPIO_WritePin(LCD_D5_PORT, LCD_D5_PIN, (data & 0x02));
	HAL_GPIO_WritePin(LCD_D6_PORT, LCD_D6_PIN, (data & 0x04));
	HAL_GPIO_WritePin(LCD_D7_PORT, LCD_D7_PIN, (data & 0x08));
	LCD_E_LOW;
}
```

```
void lcdWriteByte(uint8_t data) {
        lcdSendHalf(data >> 4);
        lcdSendHalf(data);
        HAL_Delay(1);
}

void lcdWriteCmd(uint8_t cmd) {
        LCD_RS_LOW;
        lcdWriteByte(cmd);
}

void lcdChar(char data) {
        LCD_RS_HIGH;
        lcdWriteByte(data);
}

void lcdInit(void) {
        HAL_Delay(15);

        LCD_E_LOW;
        LCD_RS_LOW;

        lcdSendHalf(0x03);
        HAL_Delay(5);
        lcdSendHalf(0x03);
        HAL_Delay(5);
        lcdSendHalf(0x03);
        HAL_Delay(5);
        lcdSendHalf(0x02);
        HAL_Delay(5);

        lcdWriteCmd( LCD_FUNC | LCD_4_BIT | LCDC_TWO_LINE | LCDC_FONT_5x7);
        HAL_Delay(5);
        lcdWriteCmd( LCD_ONOFF | LCD_DISP_ON);
        HAL_Delay(5);
        lcdWriteCmd( LCD_CLEAR);
        HAL_Delay(5);
        lcdWriteCmd( LCDC_ENTRY_MODE | LCD_EM_SHIFT_CURSOR | LCD_EM_RIGHT);
```

```
        HAL_Delay(5);
}

void lcdLocate(uint8_t x, uint8_t y) {

        switch (y) {
        case 0:
                lcdWriteCmd( LCDC_SET_DDRAM | (LCD_LINE1 + x));
                break;

        case 1:
                lcdWriteCmd( LCDC_SET_DDRAM | (LCD_LINE2 + x));
                break;
        case 2:
                lcdWriteCmd( LCDC_SET_DDRAM | (LCD_LINE3 + (x - 12)));
                break;
        case 3:
                lcdWriteCmd( LCDC_SET_DDRAM | (LCD_LINE4 + (x - 12)));
                break;
        }

}

void lcdStr(char *text) {
        while (*text)
                lcdChar(*text++);
}

void lcdInt(int data){

        char buffer[20];

        sprintf(buffer, "%d", data);

        lcdStr(buffer);

}
```

Kod programu, plik interrupts.c(obsługa przerwania):

```
#include "interrupts.h"

void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size) {

	if (huart->Instance == USART1) {

		HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 1);
		flagReceived = 1;

		HAL_UARTEx_ReceiveToIdle_DMA(&huart1, receiveBuffer, BUFFER_SIZE);
	}
}
```

4.Wnioski

Układ testowy działa poprawnie i spełnia założenia- umożliwia identyfikację i odczyt mocy sygnału z dwóch tagów.

Wykonał: Bartosz Pracz