

# Control of RF Components within a Passive RF Tracking System

Jack Heeks (2111738) – BEng Embedded Electronic Systems

, School of Applied Computing, Faculty of Architecture, Computing and Engineering

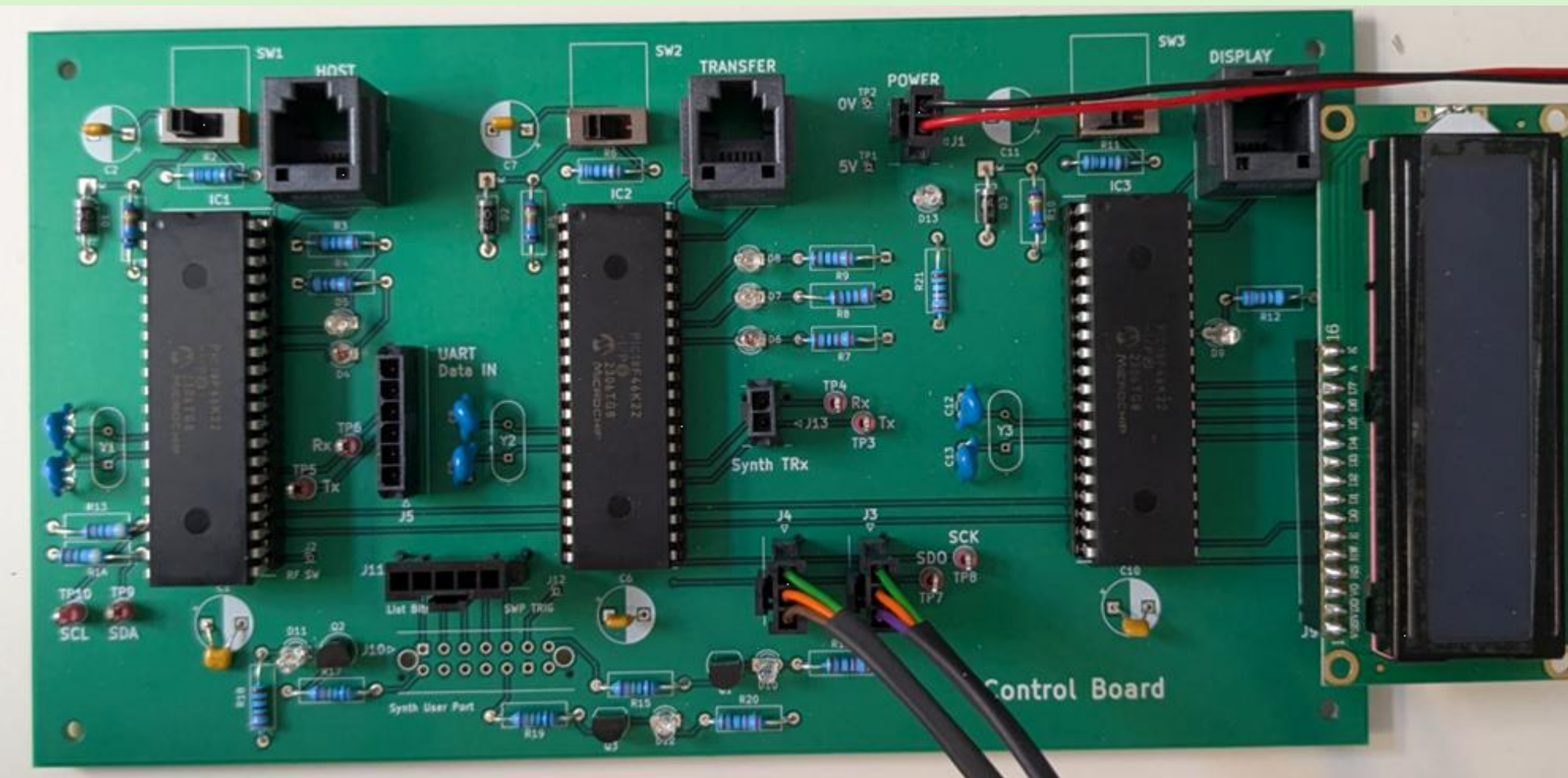
University of Wales Trinity St. David, SA1 Waterfront Campus, IQ Building, Swansea SA1 8EW | [2111738@student.uwtsd.ac.uk](mailto:2111738@student.uwtsd.ac.uk)

## 1. Introduction

Threat Detection and Countermeasures (TDaC) Group's Maritime Countermeasures Team has been developing a passive Radio Frequency (RF) tracking system for use in a variety of trials. The aim of which will be to obtain the RF transmissions for use in wider ongoing research through observation and tracking of RF sources; allowing tests to be performed and studies carried out during further trials.

## 2. Aim

The goal of this project is to develop the accuracy, reliability, and efficiency of the passive RF tracking system by optimizing the control of the RF components within the system. This will involve designing, simulating, and testing the system to develop the prototype.



Populated PCB fabricated for Prototype System

## 3. Method

MPLAB X IDE and XC8 C language compiler were chosen to write the programs as the PICs and provided Real In-Circuit Emulator device are Microchip products, Proteus will be used to simulate the code during development as this software, like the ICE device and MPLAB, is freely available for use during this project. From this the PCB designs will be carried out on KiCAD for the same reasons and once the fabrication is complete the prototype will be tested in stages building up to a full system test.

## 4. Code Development

Throughout the project various core data processing functions were developed along with Interrupt Service Routines to carry out the data transfers and byte handling. The functions include identification and initiation of data transfers; all the interrupts carry out byte transmission and reception.

```
PIE1bits.TX1IE = 1;           // enable the Tx Interrupt
TXREG1 = synthCmd[synthPtr];  // Send first character
synthCmd[synthPtr] = 0x00;
synthPtr++;

void UARTTxISR(void) {
    if (synthCmd[synthPtr] == '\0') { // check for NULL character, if not...
        PIE1bits.TX1IE = 0;         // disable the interrupt
        synthPtr = 0;               // ...and reset the pointer
    }
    else {
        TXREG1 = synthCmd[synthPtr]; // put the next character in the transmit buffer
        synthCmd[synthPtr] = 0x00;   // clear the element
        synthPtr++;                 // increment the pointer
    }
}
```

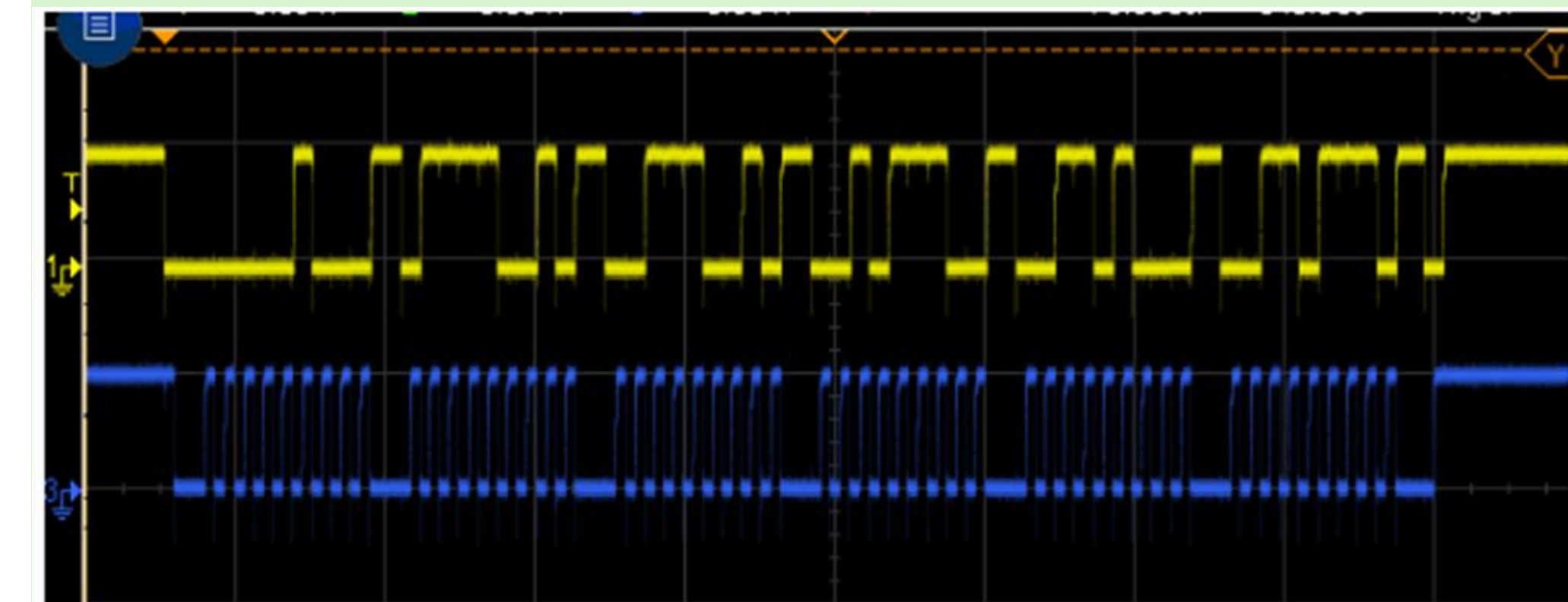
## 5. PCB Design

The PCB was designed on KiCAD using the application notes on the datasheets. As such I included components like powerline capacitors for supply stabilisation and noise mitigation ensuring the data transmissions are not affected by noise in the microcontroller. Similarly, oscillator crystals have been utilised to ensure a stable and accurate clock signal is provided for baud rate generation, this is important for the asynchronous communication protocols (RS232) as the microcontroller closely matches the desired baud rate.

## 6. Prototype Testing

The functionality of the prototype was tested in stages building up the capability slowly and making changes where necessary, the aim of testing was to establish the capability and reliability of the data transmission throughout the PCB. The main initial testing revolved

around ensuring the interrupts carrying out the data transmissions were reliable and accurate during operation. Each PIC was validated as functioning correctly in isolation before the system is slowly built up and tested further.



I<sup>2</sup>C Bus Transmission from Host PIC and Transfer and Display PICs

## 7. Future Improvements

- Power management PCB needs fully designing and testing
- Surface mount and remove unnecessary components to reduce footprint
- Device data returned to the system operator along data chain
- PCBs designed to Internationally accepted standards
- Code written in C++ and adhering to International Standards and accepted best practices
- Likelihood is circuit and code would be reduced to a single programme for a single controlling PIC

## 8. Conclusion

If the prototype was to be further developed into a production unit, the circuit would be changed to a single PIC programmed with all the necessary functionalities of the prototype circuit. The complexity of the code would be lessened by removing the requirement for inter-PIC communication as the single PIC would receive the user commands and send the instructions to the devices directly. Each element of the product would have to be designed and fabricated in line with all the International Standards required for production.

## References

1. thepiyushguptamarketing, 'Pic Microcontroller Guide: Features, Applications & Uses'. Accessed: Nov. 01, 2024. [Online]. Available: <https://fpgainsights.com/fpga/pic-microcontroller/>
2. 'Kumar - Embedded Communications Protocols and Internet of .pdf'. Accessed: Oct. 31, 2024. [Online]. Available: <http://www.gpcet.ac.in/wp-content/uploads/2018/08/UNIT-V.pdf>
3. 'PCB Layout Design - ESP32-S2 - — ESP Hardware Design Guidelines latest documentation'. Accessed: Nov. 09, 2024. [Online]. Available: <https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32s2/pcb-layout-design.html#uart>
4. 'Advantages of PIC 18 Series over PIC 16 series', NBCAFE. Accessed: Nov. 08, 2024. [Online]. Available: <https://www.nbcafe.in/advantages-of-pic-18-series-over-pic-16-series/>