JAMES COOK UNIVERSITY

COLLEGE OF SCIENCE & ENGINEERING

CC3501

Electrical and Electronics Engineering

Technical Report – Robotics

Ethan Waters | 13615232

Lachlan Pryce | 12618569

Bachelor of Engineering with Honours
(Electrical and Electronics Engineering)

1st of November 2023

Contents

1. Introduction

Living in rural and remote areas often comes with challenges related to accessing specialised medical care, lengthy response times, and the significant travel costs involved. Simultaneously, industrial operations can be hazardous and detrimental to human operators. However, recent developments in robotics offer promising solutions to these critical issues, demonstrating effectiveness in enhancing productivity and overall safety. The integration of human-robot interaction through teleoperation realises the possibility of performing safety critical medical and industrial operations in remote locations which lack the required expertise locally. An intuitive teleoperation approach involves using the posed movements of the user's body as a control input, with the robot responding in real-time. With assistance from the 6-axis robotic arm “NED” produced by Niyro, a prototype of this innovative teleoperation concept can be demonstrated.

2. Solution 1 – Sensor Glove

The principal solution utilised four separate modular and interchangeable printed circuit boards (PCBs) containing an Inertial Measurement Unit (IMU) that could provide orientation data. The four boards were to be strategically placed on the operator to best coincide with the components of the robotic arm being hand/link 4, forearm/link 3, upper arm/link 2 and chest/link 1, as demonstrated in Figure 1. The modularity ensures that any given board can be replaced with the spare in case of system failure.

Figure 1: Placement of sensors was selected to best match NEDs’ physical characteristics.

The PCBs were to employ CAN bus to collate the orientation data, then subsequently process and transmit the data to a computer from the master node. The PC will then act as the primary access point for communicating with and instructing NED. This is facilitated by a python script that utilises a documented library for NED and can then utilise the received orientational data to adjust the joints by a specified angle and direction, thereby mimicking human arm movement. This overall approach can be abstracted into a flow diagram seen in Figure 2.

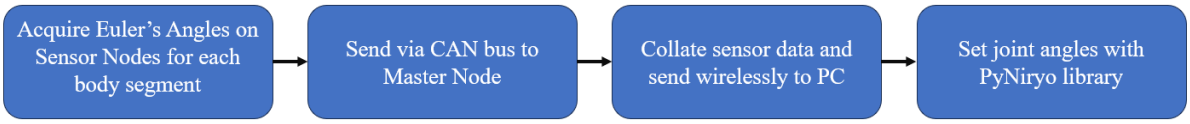


Figure 2: High level approach to arm control with sensor glove.

2.1. Hardware Design

All hardware selections were measured against performance, cost, documentation/support, and availability constraints. The complete schematic can be seen at the of this document.

2.1.1 Inertial Measurement Unit - LSM9DS1

System Requirement Satisfied: Determine orientational data in real time. Provide low pass filtering to orientational data.

A variety of IMU's were investigated as potential suitors. The market contained many high performance 6-DoF (degrees of freedom) IMU's however these inherently contained only an accelerometer and gyroscope. Yaw readings require a calibrated magnetometer to avoid the problematic yaw drift commonly experienced in IMU's. 6-DoF sensors were briefly considered (Bosch BHI380 & BMI270) with an external magnetometer (Bosch BMM350) however many of the magnetometers operate as a I²C slave to accelerometer/gyroscope, thereby limiting the achievable data output rate. With this constraint in mind, the 6-DoF sensors were ruled out. The choice of 9-DoF IMU's was limited by cost and availability. The 9-DoF Bosch BNO055 was the primary choice as it had excellent performance characteristics but was expensive and unavailable through the part provider JLCPCB. Instead, the LSM9DS1 was selected and preordered due to its availability, sufficient documentation, satisfactory performance characteristics and cost. The LSM9DS1's inclusion of a low-pass filter was a point of interest as it enhances data reliability which is critical in robotics applications.

LSM9DS1 datasheet and existing breakout board designs were utilised in the design process to develop the schematic outlined in Figure 3. Low ESR decoupling capacitors were added to the power inputs to improve voltage stability and reduce noise. Headers were also added for fundamental control pins (SCL/SPC, MOSI, MISO and CS) and auxiliary interrupt/data ready pins. This design choice allowed for greater redundancy which proved beneficial.

In a design error, the CS_M and CS_A/G pins were connected. During SPI communication the CS pin was set low to commence communication, which resulted in only the accelerometer/gyroscope being accessible. Consequently, magnetometer data could not be acquired through SPI with the CS tied configuration. This issue was circumvented by successfully implementing I²C through the redundancy header pins that corresponded to SDA and SCL on the chosen microprocessors and including 3.8 kΩ external pull up resistors, see the referenced I²C specification [1].

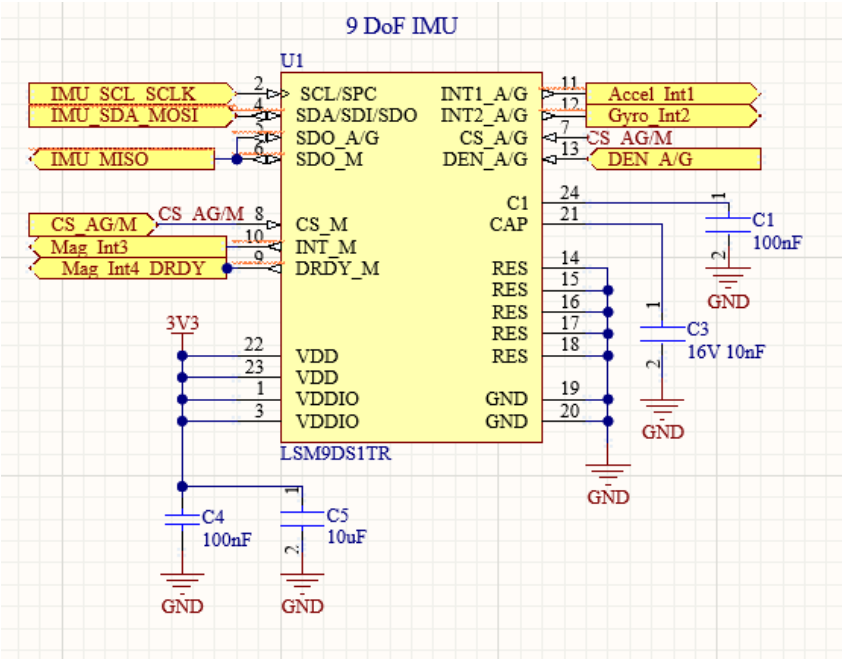


Figure 3: LSM9DS1 schematic.

2.1.2 Primary Microcontroller - RP2040

System Requirements Satisfied: Compute sensor fusion algorithms. Capacity to communicate with IMU.

The RP2040 was selected as the microcontroller to interface with the LSM9DS1 IMU. The RP2040 was relatively cheap in comparison to other microcontrollers, provided excellent documentation, software development kits in Python or C/C++ and was readily available through JLCPCB. The RP2040 supports two separate SPI channels for simultaneous communications with both the IMU and required Controller Area Network (CAN) interface controller (MCP2515) and two separate I²C buses for redundancy. Furthermore, the RP2040's debugging capabilities stand out as it can be easily debugged with a Picoprobe that utilises UART, providing a robust framework for development and troubleshooting.

Power and data transmission were provided by micro-USB with decoupling capacitors to stabilise the supply and a voltage regulator to reduce 5v to the required 3.3v of the RP2040. External flash memory was incorporated to improve the debugging processing and Serial Wire Debug (SWD) was incorporated as a redundancy for Picoprobe failure. Furthermore, UART communication was implemented between the RP2040 and the secondary microcontroller (ESP32-S3) as an alternative form of communication in the event of CAN bus failure.

2.1.3 CAN Bus Network

System Requirements Satisfied: Multi-master off board communication.

CAN bus was the chosen protocol for inter board communication. Its multi-master architecture enables integration of multiple devices and offers reliable data transmission, making it ideal for safety-critical systems, see referenced CAN bus specification [2]. The protocol's differential signalling minimises electromagnetic interference, enhancing data integrity. Additionally, CAN provides efficient error detection and correction mechanisms, contributing to system stability. The implementation of message priority levels to prevent data collisions on a two wire twisted pair allows CAN to be implemented asynchronously, providing the benefit of concurrent communication to multiple nodes when required.

Typically, CAN bus is implemented with 120Ω characteristic impedance wires and terminating resistors. However, procurement of these components was not able to be achieved at a reasonable cost,

and instead the system was designed to utilise 100Ω characteristic impedance wire stripped from CAT6 Ethernet cables. Recycling CAT6 Ethernet cables proved to be an effective solution, however, upon receiving the fabricated boards, the desired terminating resistors were no longer available. To approximate the 100Ω termination, several resistors were placed in parallel, resulting in an impedance of 130Ω. While it was feasible to add more resistors for a closer approximation, practical considerations, including physical space limitations and the risk of short circuits, led to the decision to maintain the 130Ω termination. This significantly reduced the attainable data transfer speeds to 10Kbits/s but was still sufficient to perform adequately.

CAN Controller Circuit – MCP2515

The RP2040 does not contain a peripheral to support CAN bus communications. Subsequently, a CAN controller was required to interface with the RP2040 through SPI, see the referenced SPI specification [3]. The MCP2515 was selected due to its extensive use in existing designs and the availability of a reputable open-source library compatible with the RP2040 and Pico SDK.

An external crystal oscillator timing circuit denoted by ‘Y1’, ‘C8’ and ‘C9’ in Figure D was designed to achieve 8.00 MHZ oscillation frequency required for operation. Although it is common to include a series resistor to reduce the drive strength of the oscillator circuit and current consumption, the MCP2515 documentation outlined that a typical design with the MCP2515 would not include the series resistor. The documentation of the chosen crystal (ABM3B-8.000MHZ-B2-T) confirmed that the MCP2515 would not exceed the rated power specifications. The capacitor values for C8 and C9 were chosen inline with the crystal documentation to achieve the desired 8MHz output. The capacitance of the OSC1 and OSC2 pins were considered to be zero during the process of selecting capacitor values as the Thevenin equivalent circuit for the pins outlined in the MCP2515 documentation did not indicate a capacitance.

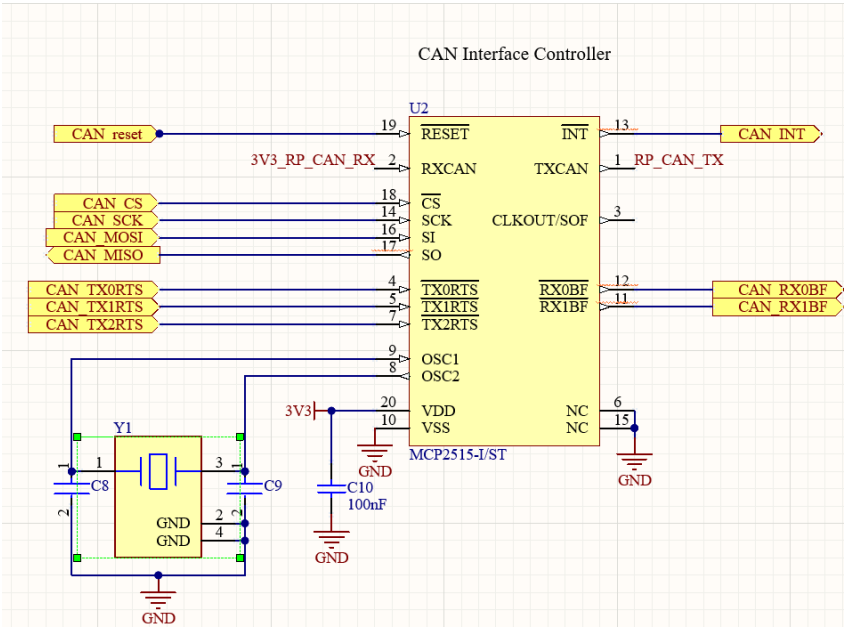


Figure 4: MCP2515 CAN bus interface schematic with timing circuit.

Transceiver – MCP2551

CAN bus communication requires a CAN Transceiver, irrespective of a microcontrollers internal or external integration of a CAN bus controller peripheral. The low cost and compatibility of the MCP2551

with the MCP2515 as demonstrated through previous designs were the determining factors for utilising the MCP2551. The datasheet and existing designs were used as a reference to create the schematic and select component values. Shown in Figure E, header pins were incorporated to provide a means of terminating the CAN bus on any board if required, ensuring modularity and easy debugging. The CAN lines implemented several capacitors to reduce noise and transient voltage suppressor diodes to prevent surge damage to the MCP2551. Notably, the MCP2551 required 5v input, resulting in logical outputs that exceed the rated voltage input for the MCP2515 CAN controller. Consequently, a logic shifter was implemented to facilitate appropriate communication between the MCP2551 and the MCP2515 CAN controller, preventing any possible damage. Lastly, the resistor at the RS pin dictated the slew rate of the MCP2551 and was set in line with the datasheet to achieve the highest slew rate within the recommended bounds.

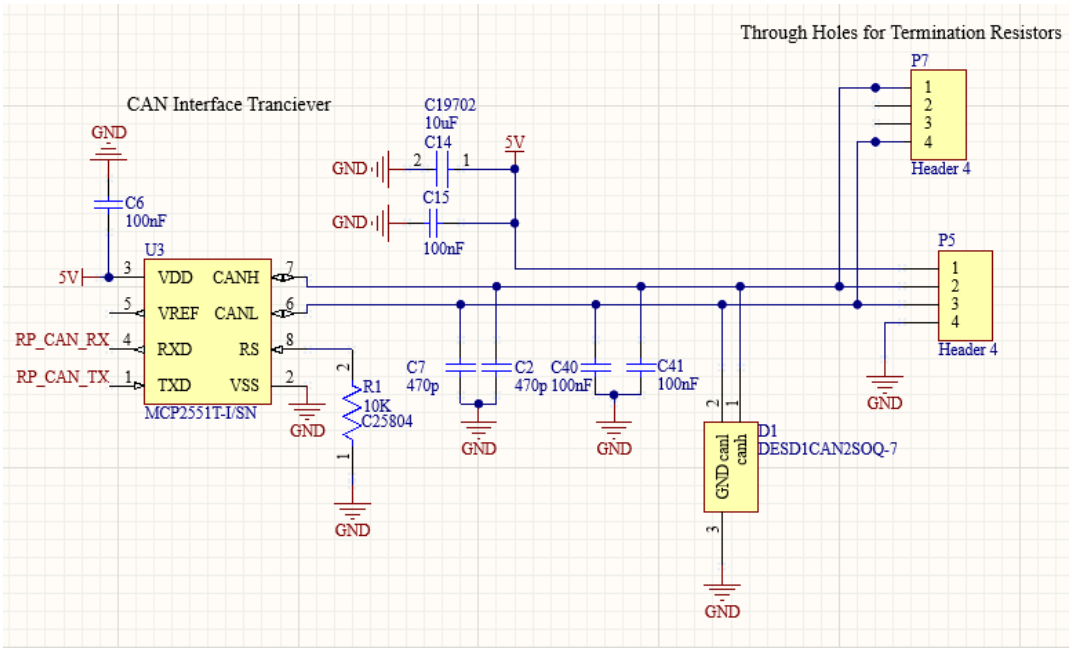


Figure 5: MCP2551 CAN transceiver schematic.

2.1.4 Secondary Microcontroller – ESP32-S3-Wroom-1

System Requirements Satisfied: Support for wireless communications.

One of the overall system objectives was to gather pitch, roll, and yaw data from all nodes and transmit it wirelessly to the PC hub. To accomplish this, Bluetooth or Wi-Fi peripherals were necessary, but the RP2040 lacked native support for both. While exploring the option of using pre-packaged embedded ICs to enable Bluetooth or Wi-Fi integration with the RP2040, it became evident that the cost of available components from the chosen manufacturer were excessive and couldn't be justified. Consequently, the ESP32-S3-Wroom-1 with built-in support for Wi-Fi or Bluetooth and TWAI (CAN bus) was selected to be utilised in conjunction with the RP2040 as a more cost-effective solution. This choice not only reduced expenses but also offered added flexibility in case of RP2040-related issues or the need for additional processing capabilities. The possibility of completely replacing the RP2040 with the ESP32-S3-Wroom-1 was considered, however decided against due to the familiarity of the RP2040, its associated development framework and lack of familiarity with the ESP32-S3-Wroom-1. The design choice of implementing both significantly increased flexibility and reduced risk for a cost that would otherwise be incurred through the implementation of a Wi-Fi peripheral. Specifically, the ESP32-S3-WROOM-1-N16R2 was utilised because of its availability, internal inclusion of a crystal oscillator and 16MB of writable flash memory.

ESP32-S3-Wroom-1-N16R2, hereafter referred to as ESP32, utilised the same CAN transceiver system previously discussed for the RP2040. No CAN controller was required for the ESP32 as any GPIO pins can access the internal CAN bus peripheral. Notably, the documentation and other ESP32 resources refer to CAN bus as Two Wire Automotive Interface (TWAI), despite following the international standards that outline CAN bus. Furthermore, the ESP23 intentionally does not have any series resistors on the microUSB lines despite it being common practice. Typical designs outlined in the documentation specifically indicate that there should be no series resistors. Lastly, an RC timing circuit was developed for the EN pin to turn the device on as specified in Section 3.3 of the documentation.

The ESP32 hardware design, while adequate, lacked user-friendly features for entering bootloader mode or removing previously loaded code from the flash. In the absence of these features, the procedure involved grounding multiple pins simultaneously, as detailed in Section 3.3.1 of the documentation.

2.1.5 Fabrication

The PCB, outlined in Figure 6, was designed to be small and compact, as to not interfere with the naturally obtainable positions of a human arm. The ESP32-S3 chip was placed on the right side of the board with the copper under and around the Wi-Fi antenna removed to maintain signal integrity.

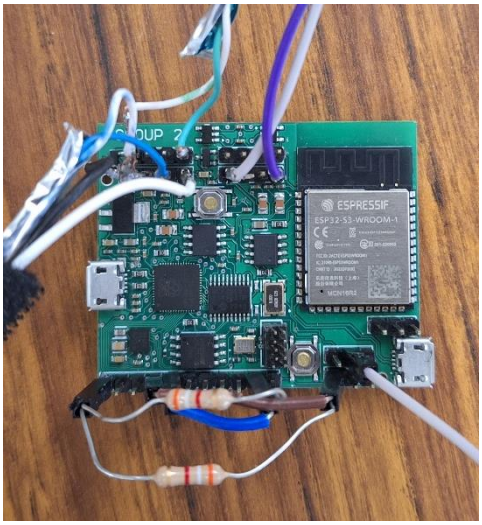


Figure 6: Fabricated PCB with added external 3.8 kΩ pull-up resistors for I²C communication.

PCB cases (Figure 7) were also 3D printed to allow the sensors to be placed on the operator and secured using Velcro straps. The PCBs were designed to be secured to the plastic case using 2mm screws, however the fabricated holes were the incorrect size and elastic bands were used as an alternative to secure the PCBs.

Figure 7: 3D printed bracket to mount PCBs and secure to operator.

2.2. Software Design

The software developed for the project comprised of C++ on the RP2040 in Visual Studio, C on the ESP32-S3 in the ESP-IDF environment and Python for socket and NED communication on the host PC. The following section will outline the software utilised on each device within the system. A software design review was conducted prior to the commencement of development, the resulting diagram can be seen in appendix (1) indicating the initial design for the software system.

2.2.1 RP2040 Embedded System Code

The overall responsibility of RP2040 code was to acquire Eulers angles when requested by the master ESP32 node and send them via CAN bus. This process is identical for each of the four RP2040s that correspond to a different segment of the body and can be seen below in Figure 8. The exact implementation of the code can be seen in the project GitHub repository [4].

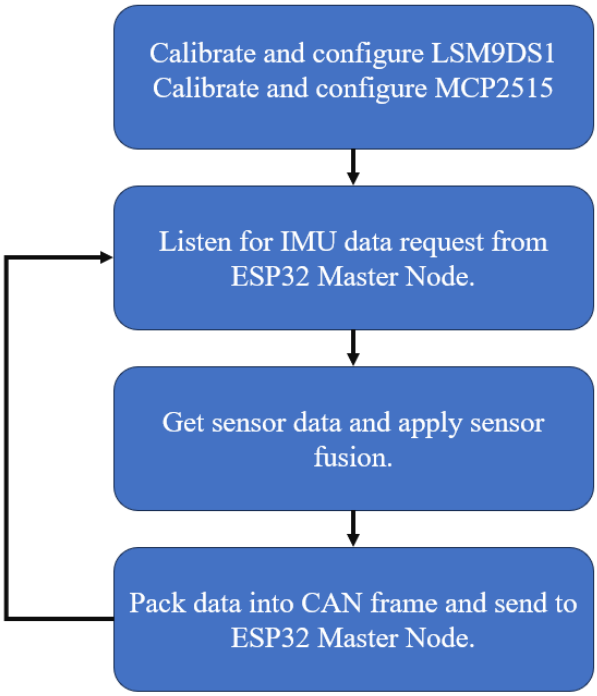


Figure 8: High level outline of RP2040 embedded operations.

IMU Code and Sensor Fusion Algorithm – LSM9DS1 Driver

The IMU code is structured as a driver, consisting of two header files and one .cpp file. This driver is designed to offer straightforward functions accessible from the main application, ensuring a high level of abstraction and hardware-specific operations' isolation. Although this driver took inspiration from an Arduino implementation, the driver was primarily developed specifically for this application. Figure 9 illustrates the core operation of the IMU code running on the RP2040.

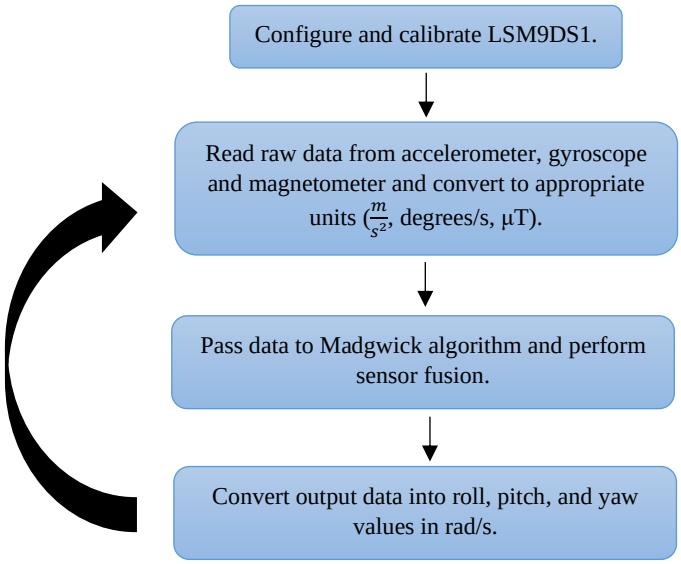


Figure 9: Block diagram of IMU embedded code operation.

With NED receiving joint angles in radians, the accelerometer, gyroscope and magnetometer were configured for high sensitivity measurement (2000 dps, 16g, 16 gauss respectively), thereby facilitating the measurement of small changes. Through testing, a medium sampling rate and bandwidth was selected to strike a balance between accuracy and response time.

Accelerometer and gyroscope calibration was required to account for inherent inaccuracies due to environmental and manufacturing factors. The calibration process for these two sensors obtained 'zero-rate' offset values that were subsequently subtracted from the final values.

Calibrated accelerometers and gyroscopes ensure stable roll and pitch data, but yaw readings can drift due to a phenomenon known as 'gyroscopic drift.' This drift results from gyroscopes offering relative, not absolute, orientation information. Magnetometers provide a constant absolute reference by measuring Earth's magnetic field, ensuring a fixed orientation. However, external magnetic interference can lead to variable readings. Magnetometer calibration determines hard and soft iron offsets to maintain a consistent magnitude. While various methods of calibrating magnetometers exist, the method of tracing in a figure of eight aimed to capture the most accurate offset parameters while maintaining system practicality. It should also be noted that for the LSM9DS1 the accelerometer and gyroscope axes are aligned, however the magnetometer axes are not. To account for this the x-axis of the magnetometer was inverted.

Initially a Kalman filter, a mathematical recursive algorithm, was implemented to further increase the output data accuracy. The Kalman filter provides optimised estimation by utilising all three sensors and comparing predicted and measured values on a recursive basis. It can perform error covariance, continually updating the error associated with predicted values. The implementation of a Kalman filter initially showed excellent results. The output orientation data was stable and subject to zero drift. Unfortunately, after prolonged motion, the output of all three Euler angles drifted considerably. It is hypothesised that the model used may have required fine tuning of gain, acceleration and magnetic rejection and sampling rate settings. Additionally, despite being addressed, it is plausible that the inverse nature of the magnetometer axes in reference to the accelerometer and gyroscope axes contributed to the unstable nature of the orientation data.

Alternative filtering systems were explored, with a focus on the Mahony and Madgwick algorithms. Both algorithms seek to determine orientation by merging sensor data and incorporating gyroscopic data, which is adjusted to account for the influence of the magnetic fields and gravity. Mahony utilises

a proportional and integral controller to correct the gyroscope bias and requires little computing capacity. In contrast, Madgwick only employs a proportional controller but achieves greater accuracy at the cost of increased computing time. It was determined that the computational trade-off for more accurate results was desirable and hence Madgwick was selected. Tests conducted using one IMU with a Madgwick filter yielded excellent results. Once calibration processes were complete the sensor response time was approximately 5-8 seconds before stable orientation data was observed. Additionally, the IMU maintained its established internal reference frame on all Euler axes.

CAN Bus Code – MCP2515 Driver

The overall implementation of the MCP2515 driver was straightforward and did not require any augmentation. The open-source library tailored for the MCP2515 and RP2040 offered a high level of abstraction and facilitated seamless CAN bus communication [5]. The constructor of MCP2515 class initialises necessary SPI communication between the RP2040 and CAN controller. Additionally, driver member functions provided the necessary functionality to configure the CAN bus mode, speed, expected oscillator input frequency. The provided functionality for sending and receiving messages conveniently checked the CAN controller registers for error flags, simplifying the debugging process. Opting to utilise this open-source library significantly reduced development time of the project.

2.2.2 ESP32-S3 Embedded System Code

The primary role of the ESP32 within the sensor glove design is to function as the master node. This encompasses the responsibility of soliciting data from the distinct RP2040 sensor nodes and maintaining data categorisation. This architectural choice ensures arm segment data is packaged consistently in the same manner, reducing the risk of error. The master node is then responsible for wirelessly transmitting the uniformly structured data via User Datagram Protocol (UDP), see reference for UDP specification [6]. UDP was preferred over TCP as it allows for easier debugging and testing, as it does not require the establishment of a connection for data transmission. Additionally, various implementations of code that utilise the transmitted data as an input could be tested without interrupting the transmission.

The initial design approach intended for the ESP32 to seamlessly interface with the Arduino IDE and its associated libraries due to their simple user-friendly nature. However, after completing the implementation of serial and UDP communication it was realised that the currently available Arduino libraries do not support the ESP-S3 line of microprocessors. This issue successfully circumvented by implementing the required functionality in C with the Espressif IoT Development Framework (ESP-IDF), despite the increased complexity and new framework.

2.2.3 PC Python Code

Although it is possible to utilize MicroPython for direct communication with the NED robotic arm from the embedded system, this approach was intentionally avoided. Instead, a Python script executed on a PC was employed, to expedite the debugging process and increase system visibility. The script employs threading to receive via User Datagram Protocol (UDP), enabling the simultaneous acquisition and utilization of data in real-time. The received data is implemented into a queue and signals to the main with an event setting flag that it is available, ensuring that is not corrupted. The Euler angles can then be passed through the butter worth filter if desired and into the desired kinematic solution before then setting the joint angles. The decision was made to perform any additional filtering and kinematic solution calculations on the PC due to its substantial processing capacity, surpassing that of the embedded system.

2.2.4 Filter Implementation

In the field of robotics, it is common practice to incorporate filters for enhancing the accuracy of IMU outputs as well as the outputs of kinematic solutions [7, 8]. During the preliminary assessments of data, we encountered notable jitteriness in the obtained data. To address this issue and enhance the overall data quality, our investigation led us to explore the implementation of filtering techniques.

After an analysis of the IMU calibration code revealed a significant coding error that contributed significantly to the observed jitteriness. Correcting this coding error resulted in a marked improvement in the data quality. However, to refine the output and ensure its stability, different filtering techniques were evaluated. This testing process involved the assessment of the IMU's inbuilt low-pass filter, the implementation of a 4th order Butterworth filter as detailed in [8], and a control scenario with no filtering. The sensor's locations were fixed during the test of each filtering approach and results were stored in separate files. These data logs are available for reference in the project's GitHub repository within "documentation".

The analysis of the Euler angle data was conducted with R and involved collecting 500 samples for each angle, subsequently subjecting them to Bartlett's Test to check the homogeneity of variances across groups. It is based on the chi-squared distribution and tests the null hypothesis that the variances are equal. The results determined that there was a statistically significant difference in the variance in the Euler angles, depending on the applied filtering method, see Table (1). Identifying this distinction prompted a comparison of the standard deviations associated with each filtering process to quantify the variance, see Table (2).

Table (1): Bartlett's Test to evaluate homogeneity of variances for filters.

	Df	K-Squared	P-Value
Yaw	2	355.85	<2e-16
Pitch	2	708.24	<2e-16
Roll	2	11252	<2e-16

Table (2): Standard deviation of Euler angles for different filters

Filter	Yaw	Pitch	Roll
Butterworth	0.003558	0.00066	0.001801
IMU Lowpass	0.008409	0.002335	1.395928
No Filter	0.00529	0.001244	0.002617

The analysis of the results reveals important insights into the impact of filtering methods on the observed angles. Notably, the implementation of the IMU low-pass filter appears to have a detrimental effect on all observed angles, with a particularly pronounced negative influence on roll. In contrast, the Butterworth filter successfully achieves its intended goal of reducing output variation, consequently mitigating jitter. However, it is important to note that this enhanced stability comes at the expense of responsiveness and range of motion, but these factors were not quantified in the tests. The Butterworth filter can easily be applied within the Python code on the PC, to depending on the desired output.

2.2.5 Kinematic Solution

An inverse kinematic solution was developed for the robotic arm in this project. Typically, inverse kinematics calculate the arm's position and orientation based on the XYZ coordinates and pitch-roll-yaw angles of the end effector, however, this can lead to lead to multiple solutions. Instead, joint angles were derived from first principles with the Denavit-Hartenberg notation for Ned2 as a visual reference, see Figure 10. Rotational matrices for each link (robot arm segment) were developed and then the relative rotational matrix between any two consecutive links calculated such that the angles between the links can then be derived. The joint angles for rotating a segment utilised a Euler angle directly, see example calculations below where R, P and Y correspond to roll pitch and yaw for their relative segments.

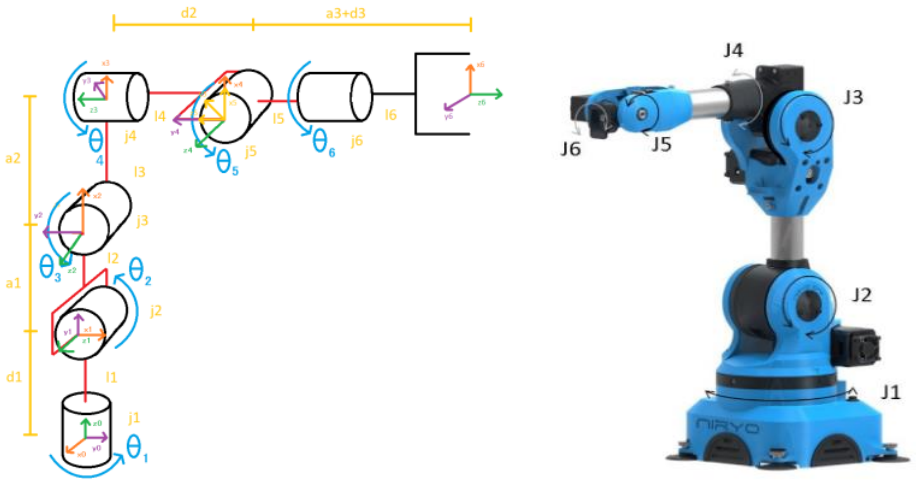


Figure 10: Denavit-Hartenberg notation for Ned2 from [9]

$$Forearm\ Rotation\ (Joint\ 4)$$

$$\Theta_4 = R_F$$

$$Forearm\ Tilt\ (Joint\ 3)$$

$$Determine\ link\ rotational\ matrix$$

$$L_{3X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(R_F) & -\sin(R_F) \\ 0 & \sin(R_F) & \cos(R_F) \end{bmatrix}$$

$$L_{3Y} = \begin{bmatrix} \cos(P_F) & 0 & \sin(P_F) \\ 0 & 1 & 0 \\ -\sin(P_F) & 0 & \cos(P_F) \end{bmatrix}$$

$$L_{3Z} = \begin{bmatrix} \cos(Y_F) & -\sin(Y_F) & 0 \\ \sin(Y_F) & \cos(Y_F) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$L_3 = L_{3X} L_{3Y} L_{3Z}$$

$$Repeat\ for\ Link\ 2$$

$$Determine\ relative\ rotational\ matrix$$

$$L_{32} = L_3^T L_2$$

$$\Theta_3 = \arccos\left(\frac{tr(L_{32}) - 1}{2}\right)$$

Regrettably, the testing of this kinematic approach remained incomplete, and its overall effectiveness remains uncertain. This outcome can be attributed to the difficulty encountered in aligning the sensor glove and robot arm to the same reference frame, given the constraint of the small, allocated development time. Consequently, the testing process became unintuitive, and the limitations of the development timeframe further compounded the difficulties, rendering it infeasible to allocate additional resources for comprehensive testing. As a result, the ultimate performance and suitability of the kinematic solution have yet to be ascertained.

3. Solution 2 – Image Processing

A viable solution for robotic arm movement involves the utilization of image processing to assess arm dynamics and determine joint angles. The development of this approach maintained the configuration of the Niryo NED robotic arm and PC, replacing the custom sensor glove with off-the-shelf hardware. Specifically, it leveraged the capabilities of a Raspberry Pi 4 and a PiCamera to facilitate image processing and the determination of joint angles. The utilisation of off the shelf hardware was primarily investigated to eliminate the risk associated with manufacturing and procurement of materials.

3.1. Software Design

The fundamental method employed in this system leverages basic image thresholding to delineate distinct objects within a stream of frames. To ensure efficient and coherent video capture, the initialization process utilised a shared pointer. This was necessitated by the inherent limitations of the camera hardware, which could not be accessed simultaneously by multiple instances of a single class. Consequently, the approach was devised to allow individual instances of the object detection class to focus on the detection of specific objects within the visual feed.

The system incorporates an interactive calibration phase as a crucial component of the image processing pipeline. This phase serves to establish the specific characteristics and parameters employed for image thresholding. Given the system's sensitivity to environmental conditions and variations in the observed objects, it is essential to recalibrate for any alterations in the environment or the objects being detected. The core concept revolves around the recognition of a sequence of three consecutive points or objects, which are strategically positioned on the joint. These points collectively serve as reference markers, facilitating the calculation of the relative angles between the lines formed by connecting them. Extracting this angular information can then be utilised to determine the difference between the current and last position of the body. These differences can then send via UDP to the PC, making use of the same python script and hardware described in the previous solution.

This was able to successfully detect for separate objects and then calculate relative angles between the lines formed by connecting the centroids of the objects, see Figure (11). Although it is plausible to achieve control of the robot with this method, it is crude, and more sophisticated methods including machine learning, or the method outlined in section 2 should be utilised to avoid the significant noise introduced by changes in lighting.

Figure 11: Four instances of object detection and the relative angle output over socket.

4. Recommendations

The following recommendations are proposed based on the development of Solution 1.

- Standard 2.54’ pitch male headers were implemented for all CAN bus communication wiring. This decision was made inline with budget and availability constraints but proved to be problematic. During testing the CAN 5V, GND, CAN_L and CAN_H wires frequently formed poor connections and resulted in extended time spent fault finding. Fit for purpose CAN bus terminals are recommended to ensure reliable communication between boards.
- To increase the data rate of CAN bus communication and reduce reflections, the terminating resistors should be selected to match the impedance of the CAN_L and CAN_H wires. For the ethernet cable used in this solution, the correct terminating resistor values are 100 Ω. In the case of a typical CAN bus this would consist of 120 Ω impedance wire with 120 Ω terminating resistors.
- The implementation of a common reference frame between NED and the IMU’s is recommended to negate the requirement to use orientation differences to drive NEDs’ joints. This could be achieved in a multi-calibration step where the operator assumes varying positions to establish a base position, Z-axis, X-axis and Y-axis respectively.
- The LSM9DS1 was chosen for its availability and cost, but it is now considered obsolete compared to newer IMUs. Its low-cost limits performance and accuracy while making it susceptible to environmental interference. Upgrading to a more robust and accurate IMU is recommended to improve system performance.
- A correctly calibrated Kalman filter operated on a host PC could have improved the orientation data accuracy without subjecting the embedded hardware to extensive floating-point calculations. This would facilitate a higher data transfer rate as raw data can be sent across the CAN with mathematical operations performed on the host PC.
- The application of an inverse kinematic solution is required to successfully translate human positioning to that of a pose obtainable by the robot. This would enable the robot to achieve the complexity, flexibility, and adaptability of human arm movements, thereby creating a powerful tool in human-robot collaboration.
- Use of a low pass Butterworth filter demonstrated increased stability at the cost of reduced range of motion and increased settling time. Greater exploration into the use of the Butterworth filter is recommended to establish its potential in the context of IMU controlled robotics.
- The IMU’s susceptibility to noise can be decreased by improving the hardware design. Shielding, component placement, improved high-frequency noise filters and separation of circuits according to their frequency and switching levels are recommended to improve IMU performance.
- The implementation of a thorough design check process during the hardware design phase is recommended to increase the likelihood of identifying design errors such as joining CS_A/G and CS_M pins.
- To enhance the ease of resetting and flashing the ESP32 chip, it is advisable to incorporate user-friendly bootloader hardware for the ESP32-S3-WROOM-1-N16R2.
- The MCP25625 offers a combined CAN controller and CAN transceiver in the one package. It features in-built logic shifting for interfacing with both 3.3v and 5v microcontrollers and has free source C++ drivers readily available. It is recommended that the MCP25625 is used to interface with the RP2040 to reduce the components required to implement CAN bus communication.

5. Conclusion

A sensor glove-based teleoperation system for a robotic arm was developed, focusing on addressing challenges in remote medical and industrial applications. A detailed account of the hardware and software design aspects of the project is provided, offering valuable insights into the development process and justification of design choices. Hardware selection choices are justified based on performance, cost, and availability considerations. The implementation of the CAN bus for multi-master communication is discussed, highlighting its importance in realising the teleoperation concept. The software design section outlines the development of the three major code bases, the RP2040 Nodes, ESP32 Master Node and PC hub. Furthermore, it delves into the implementation of filtering and kinematic solutions to enhance the system's accuracy and stability. Overall, the project was successful demonstrating the plausibility of utilising robotics wirelessly to perform tasks.

6. References

[1] *I2C-bus specification and user manual*, N. Semiconductors, 2021 [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[2] *ISO 11898-1:2015 Controller area network (CAN)*, ISO, 2015. [Online]. Available: https://www.iso.org/standard/63648.html

[3] *SPI Block Guide V03.06*, I. Motorola, 2000. [Online]. Available: https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf

[4] E. Waters and L. Pryce. "CC3501." https://github.com/EthanWaters/CC3501 (accessed 2023).

[5] P. Adamczyk. "Pico-MCP2515." https://github.com/adamczykpiotr/pico-mcp2515 (accessed 2023).

[6] *User Datagram Protocol*, 1980. [Online]. Available: https://www.rfc-editor.org/info/rfc768

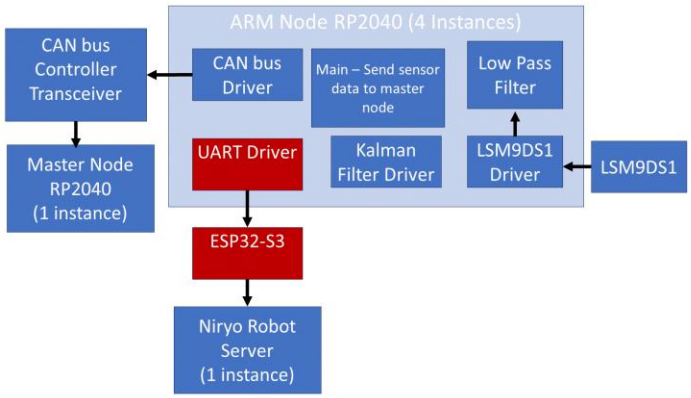
[7] Z. Fang, S. Woodford, D. Senanayake, and D. Ackland, "Conversion of Upper-Limb Inertial Measurement Unit Data to Joint Angles: A Systematic Review," (in eng), *Sensors (Basel)*, vol. 23, no. 14, Jul 19 2023, doi: 10.3390/s23146535.

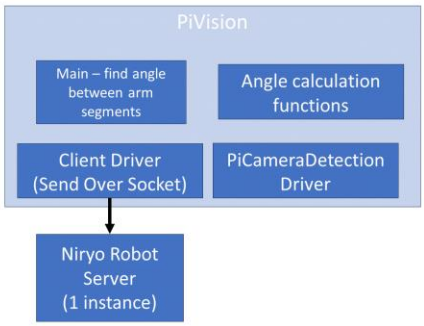
[8] M. M. Rahman, K. B. Gan, N. A. A. Aziz, A. Huong, and H. W. You, "Upper Limb Joint Angle Estimation Using Wearable IMUs and Personalized Calibration Algorithm," *Mathematics*, vol. 11, no. 4, p. 970, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/4/970.

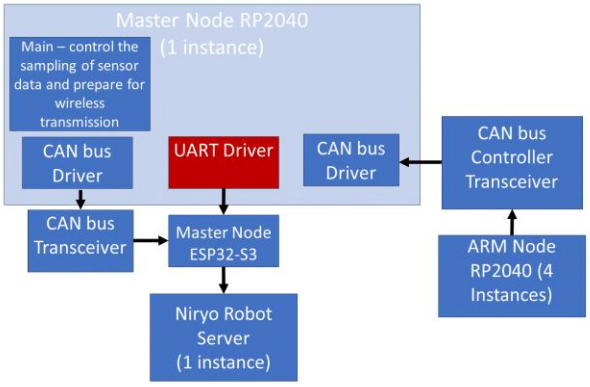
[9] S. Hyleh, "Control of Robots," Mechanical and Sustainable Engineering, Arcada University of Applied Sciences, Finland, 2023.

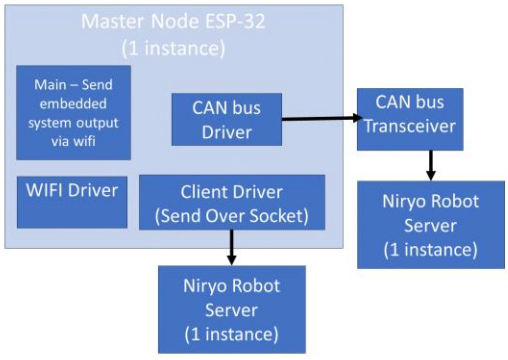
7. Appendices

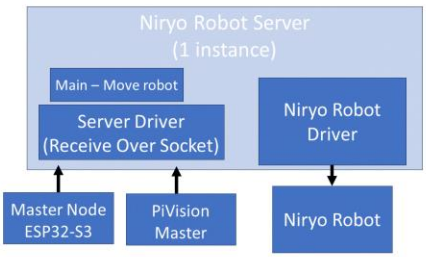
7.1. Appendix 1

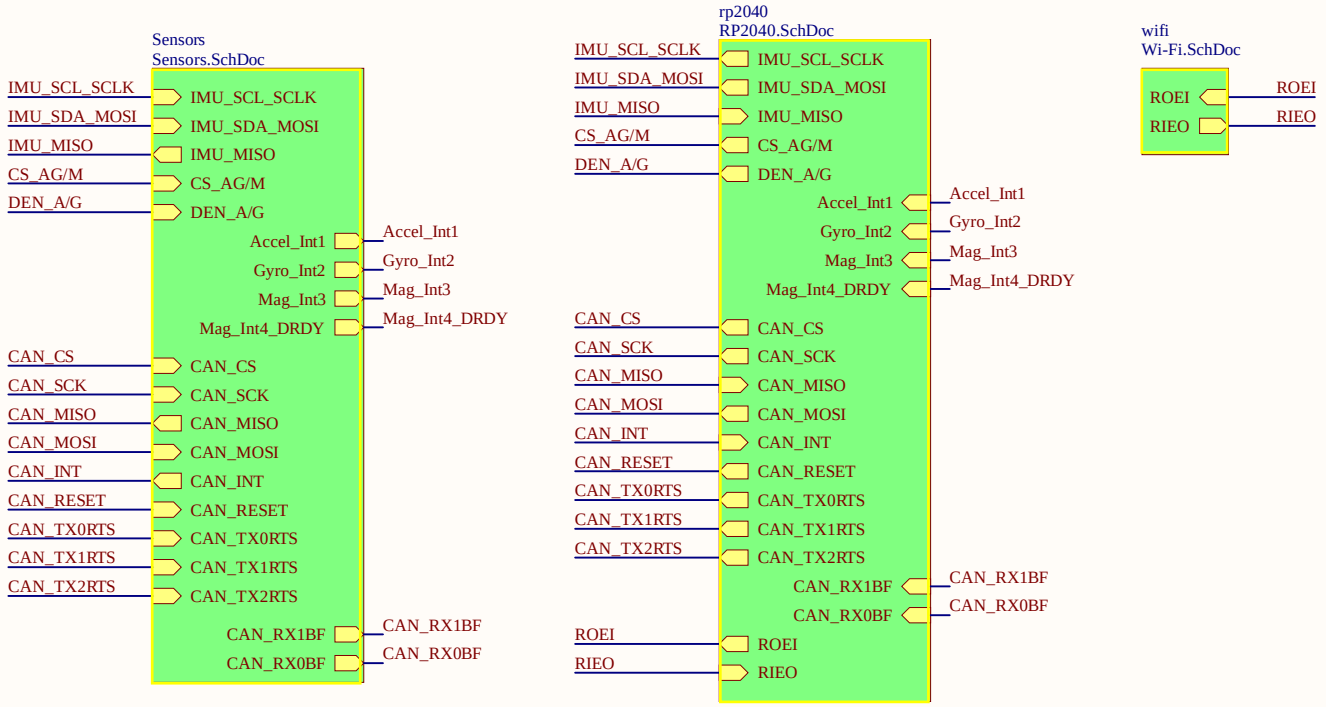




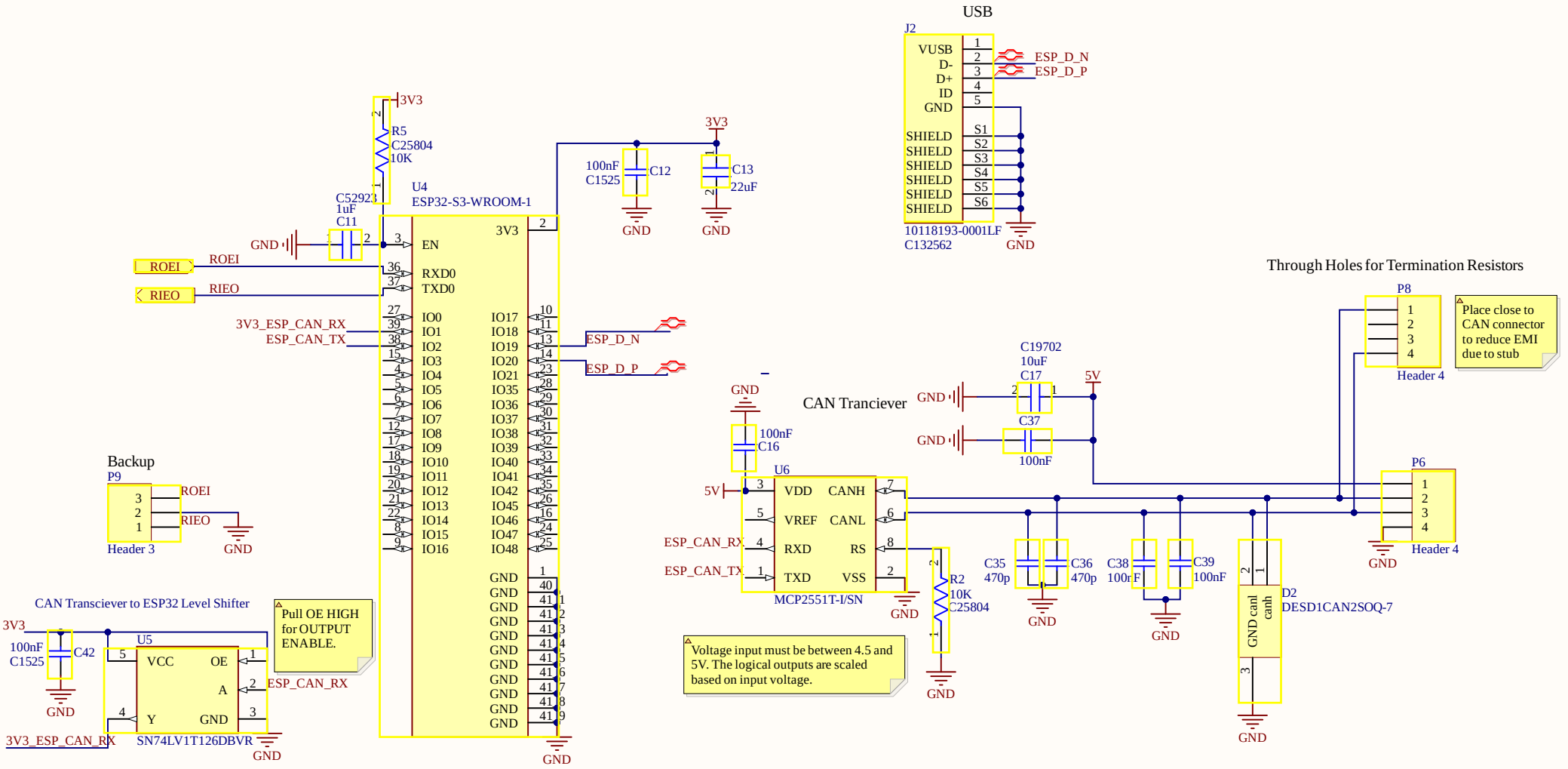








CC3501 A2 Main	Drawn by: Group 2	JAMES COOK UNIVERSITY AUSTRALIA
Sheet 1 of 4		
Date: 1/11/2023 Time: 5:11:36 PM	Revision: 0	
File: Main.SchDoc		



9.3.3.1 Down Translation

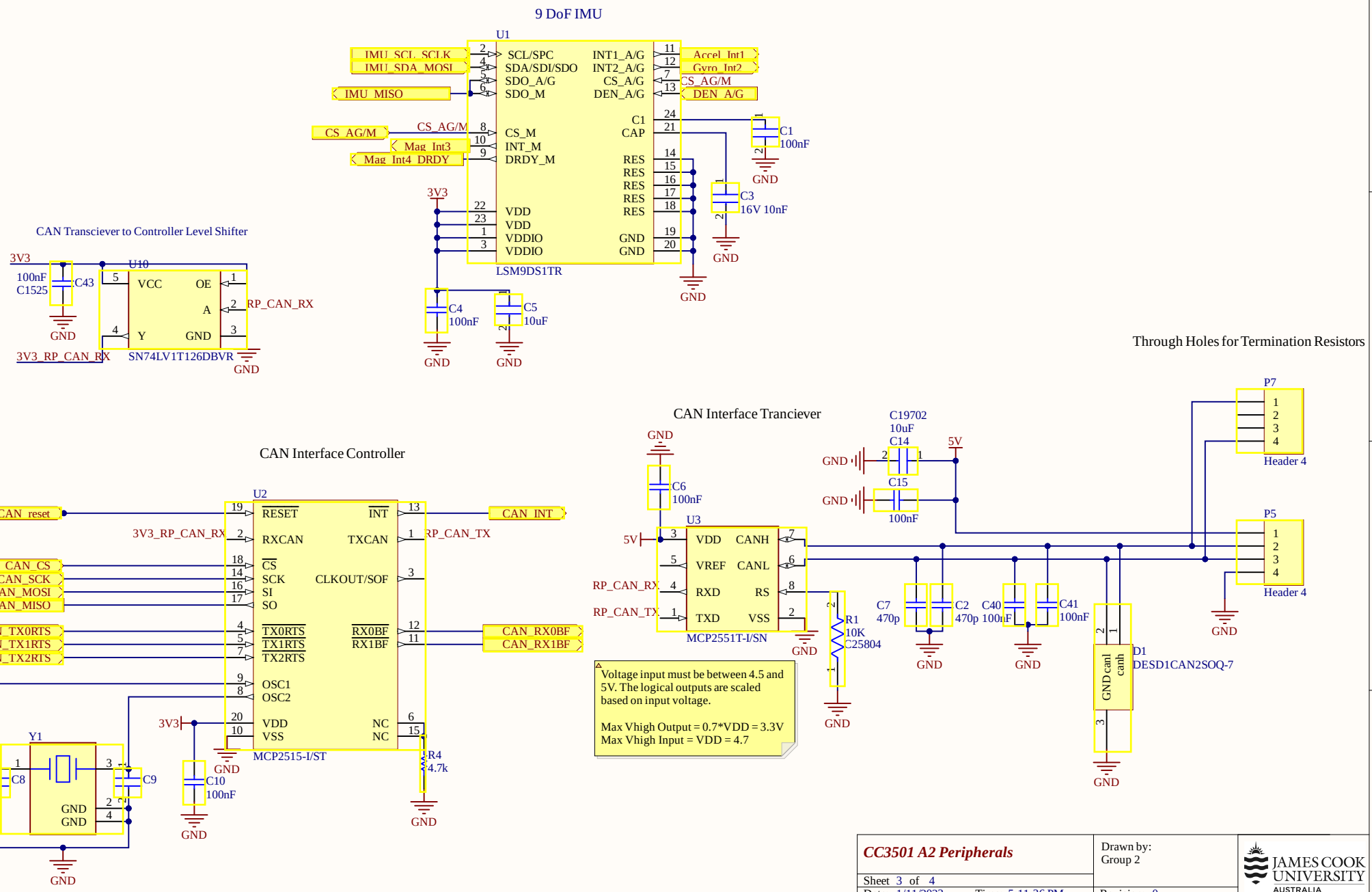
Signals can be translated down using the SN74LV1T126. The voltage applied at the V_{CC} will determine the output voltage and the input thresholds as described in the *Recommended Operating Conditions* and *Electrical Characteristics* tables.

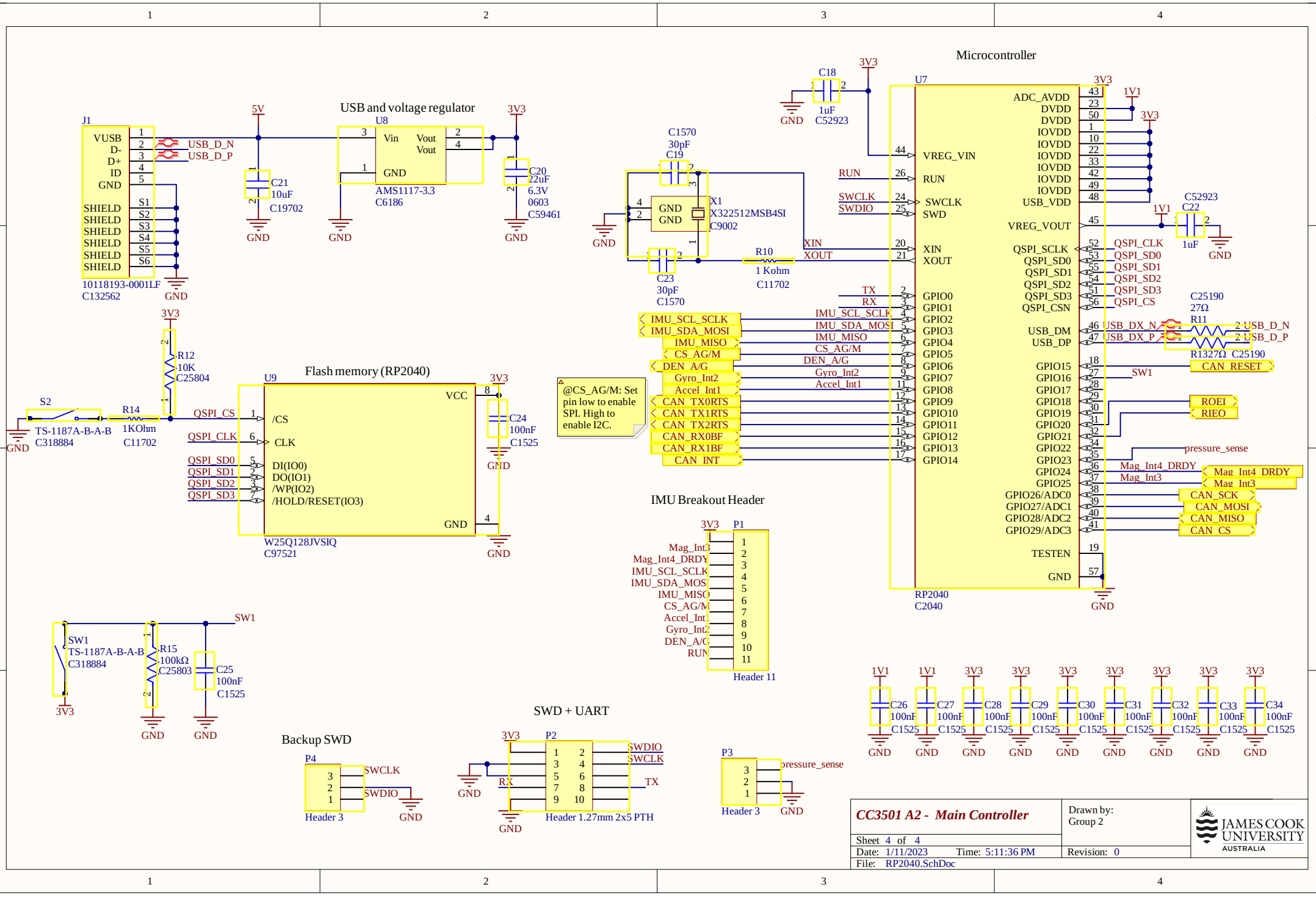
When connected to a high-impedance input, the output voltage will be approximately V_{CC} in the HIGH state, and 0 V in the LOW state. Ensure that the input signals in the HIGH state are between $V_{IH(MIN)}$ and 5.5 V, and input signals in the LOW state are lower than $V_{IL(MAX)}$ as shown in Figure 9-3.

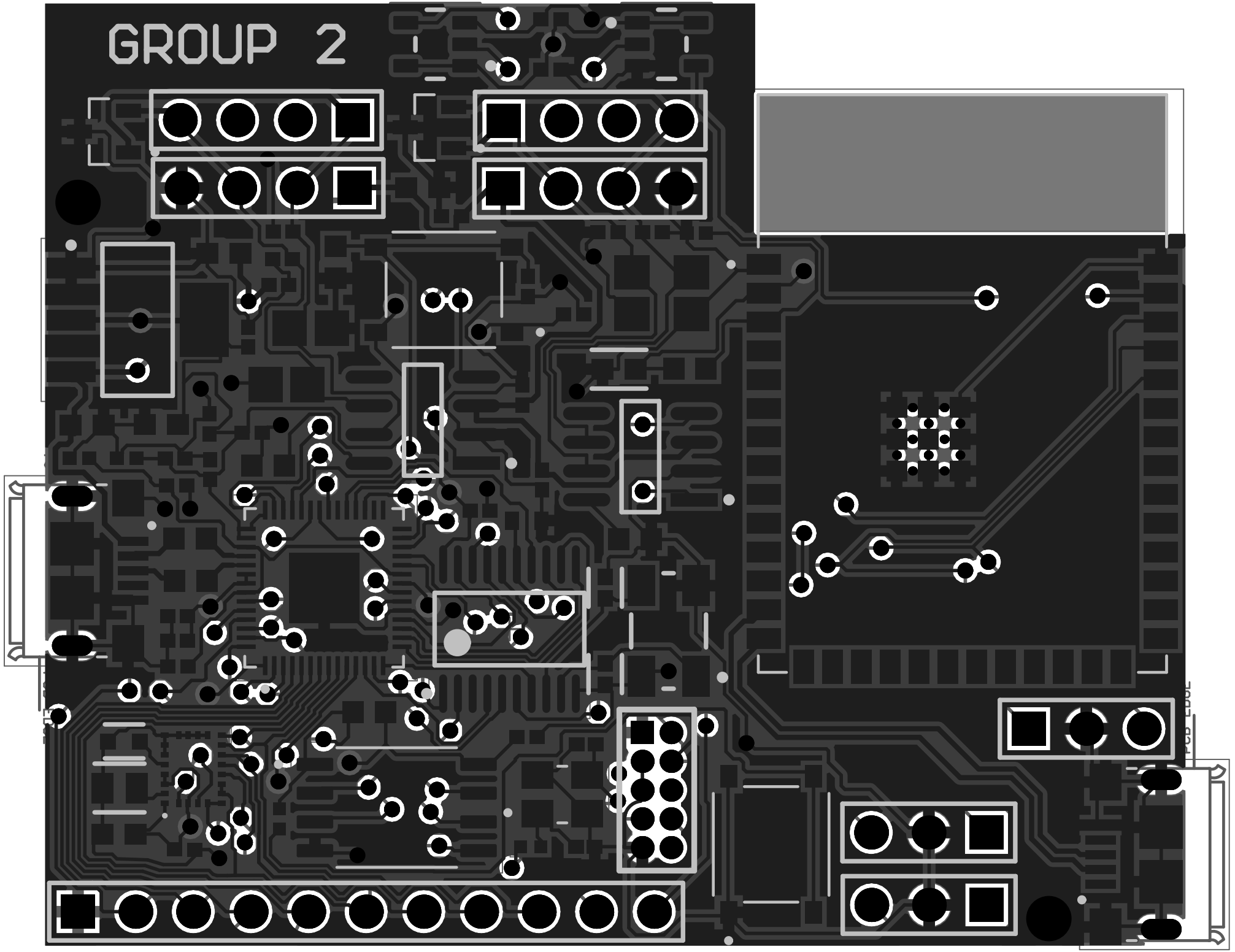
For example, standard CMOS inputs for devices operating at 5.0 V, 3.3 V or 2.5 V can be down-translated to match 1.8 V CMOS signals when operating from 1.8-V V_{CC}. See Figure 9-4.

CC3501 A2 Wi-Fi Module		Drawn by: Group 2
Sheet 2 of 4		
Date: 1/11/2023	Time: 5:11:36 PM	Revision: 0
File: Wi-Fi.SchDoc		

JAMES COOK UNIVERSITY AUSTRALIA


9 DoF IMU
U1
IMU_SCL_SCLK
IMU_SDA_MOSI
IMU_MISO
CS_AG/M
Mag_Int3
Mag_Int4_DRDY
SCL/SPC
SDA/SDI/SDO
SDO_A/G
SDO_M
CS_M
INT_M
DRDY_M
INT1_A/G
INT2_A/G
CS_A/G
DEN_A/G
C1
CAP
RES
VDD
VDDIO
GND
Accel_Int1
Gyro_Int2
DEN_A/G
LSM9DS1TR
3V3
C1
100nF
C3
16V 10nF
C4
100nF
C5
10uF
CAN Transciever to Controller Level Shifter
U10
VCC
OE
A
Y
GND
C43
100nF
C1525
RP_CAN_RX
3V3_RP_CAN_RX
SN74LV1T126DBVR
CAN Interface Controller
U2
CAN_reset
3V3_RP_CAN_RX
CAN_CS
CAN_SCK
CAN_MOSI
CAN_MISO
CAN_TX0RTS
CAN_TX1RTS
CAN_TX2RTS
RESET
RXCAN
CS
SCK
SI
SO
TX0RTS
TX1RTS
TX2RTS
OSC1
OSC2
VDD
VSS
INT
TXCAN
CLKOUT/SOF
RX0BF
RX1BF
NC
CAN_INT
RP_CAN_TX
CAN_RX0BF
CAN_RX1BF
MCP2515-I/ST
Y1
C8
C9
C10
100nF
R4
4.7k
CAN Interface Tranciever
U3
VDD
VREF
RXD
TXD
CANH
CANL
RS
VSS
RP_CAN_RX
RP_CAN_TX
MCP2551T-I/SN
C6
100nF
C19702
10uF
C14
C15
100nF
5V
R1
10K
C25804
C7
470p
C2
470p
C40
100nF
C41
100nF
D1
DESD1CAN2SOQ-7
GND canl
canh
Voltage input must be between 4.5 and 5V. The logical outputs are scaled based on input voltage.
Max Vhigh Output = 0.7*VDD = 3.3V
Max Vhigh Input = VDD = 4.7
Through Holes for Termination Resistors
P7
P5
Header 4
GND
CC3501 A2 Peripherals
Sheet 3 of 4
Date: 1/11/2023 Time: 5:11:36 PM
File: Sensors.SchDoc
Drawn by:
Group 2
Revision: 0
JAMES COOK UNIVERSITY
AUSTRALIA


Microcontroller
USB and voltage regulator
Flash memory (RP2040)
IMU Breakout Header
SWD + UART
Backup SWD
J1
10118193-0001LF
C132562
U8
AMS1117-3.3
C6186
C19702
C59461
U7
RP2040
C2040
C52923
C1570
X1
X322512MSB4SI
C9002
R10
1 Kohm
C11702
U9
W25Q128JVSIQ
C97521
R12
10K
C25804
R14
1KOhm
C11702
S2
TS-1187A-B-A-B
C318884
SW1
R15
100kΩ
C25803
C25
100nF
C1525
C24
100nF
C1525
C25190
27Ω
R11
R13270Ω
C25190
@CS_AG/M: Set pin low to enable SPI. High to enable I2C.
P1
Header 11
P2
Header 1.27mm 2x5 PTH
P3
Header 3
P4
Header 3
C18
1uF
C52923
C22
1uF
C26
C27
C28
C29
C30
C31
C32
C33
C34
100nF
C1525
CC3501 A2 - Main Controller
Drawn by:
Group 2
Sheet 4 of 4
Date: 1/11/2023 Time: 5:11:36 PM
Revision: 0
File: RP2040.SchDoc
JAMES COOK UNIVERSITY
AUSTRALIA


GROUP 2