CC3501 Weekly Report

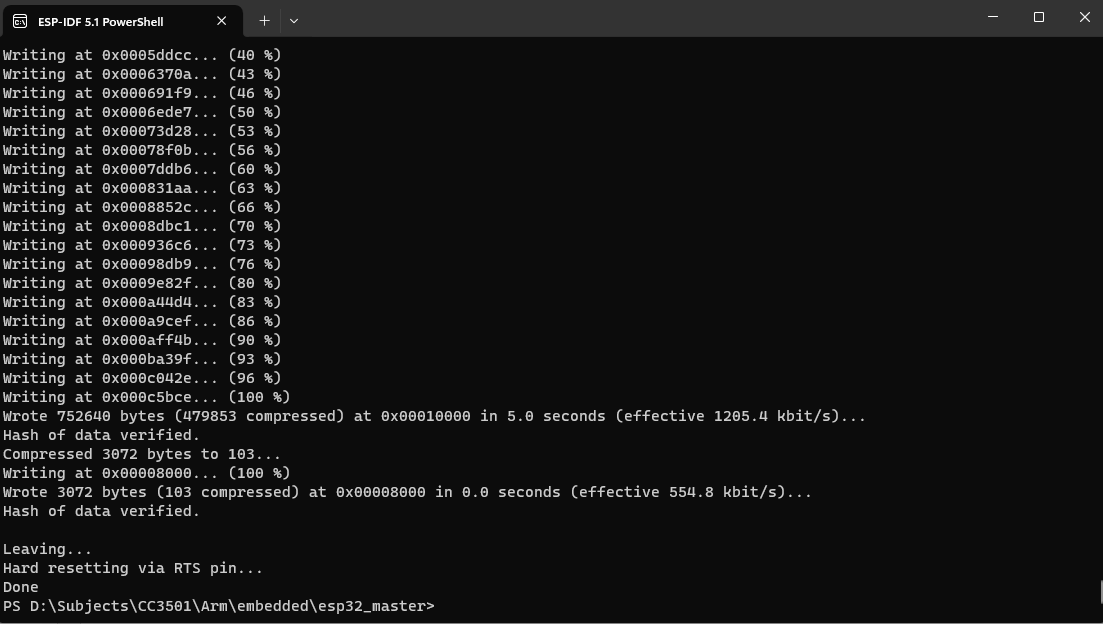
**Group number:** 2 **Team members:** Ethan Waters, Lachlan Pryce  
**Week number:** 12

**Progress this week**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Who did it?** | **What were the outcomes?** | **What did you learn?** | **Who Reviewed** |
| Create more permanent solution for can bus wires | Lachlan | more permanent solution for can bus wires |  |  |
| Fix CAN bus code for RP2040 | Ethan | Checked SPI communications and debugged with Bronson and Lachlan. Fixed a couple minor code bugs. Checked crystal. Realised I misinterpreted the reset pin requirements of CAN controller. Corrected reset pin voltage and CAN bus code for RP2040 now works. | How CAN controller MCP2515 works. | Bronson & Lachlan |
| Trouble Shoot ESP32-S3 | Ethan | Realised that COM port could not be found for esp32 because it needs to enter bootloader mode before uploading. The hardware design did not implement this to occur automatically or via a button. Requires two pins to be shorted to ground to enter bootloader mode. Then once uploaded requires a hard reset via shorting reset pin. | How to flash ESP32 correctly and what could have been added to PCB for more user-friendly outcome. |  |
| ESP32-S3 Main code | Ethan | Wrote Arduino script for socket and serial communication on ESP32-S3. However realised that current Arduino libraries for internal CAN bus controller of ESP32 chips do not reliably support the ESP32-S3.   Installed ESP-IDF development environment, frameworks and libraries. Replicated previous Arduino script with C libraries from ESP-IDF dev environment. Once working then was able to implement CAN bus with C library on ESP32. This successfully then communicated with RP2040 CAN bus code previously written. |  | Lachlan |
|  |  |  |  |  |
|  |  |  |  |  |

**Overall project tracking:** Still working on all subsystems functioning.

|  |  |
| --- | --- |
| **Week number** | **Milestones** |
| 1 | Confirm project topic and begin |
| 2 |  |
| 3 | Arm can move with an input from a socket. The input is an automated test script executed by a client to mimic the embedded system output |
| 4 | Select components & review datasheets |
| 5 | Begin schematic |
| 6 | Complete Schematic, forward to Bronson for feedback. |
| 7 | Submit complete schematic to Bronson for feedback. Complete PCB design for feedback, |
| 8 | Submit board for manufacture. |
| 9 |  |
| LR | Work on vision based movement with PI while waiting for embedded systems. Work on sensor calibration and Kalman filter code. |
|  |  |
| 11 |  |
| 12 |  |
| 13 | All subsystems working to start testing arm |
|  | Demo day Wednesday Swotvac |

This is the ESP-IDF development environment used to program the esp32. Installing the VSCODE extension proved to be time consuming so I abandoned that and was able to code fine with just this.

A computer screen shot of a black screen

Description automatically generatedreceived transmission on RP2040 from ESP32-S3