Titan Electronics System Overview

Contents

Turning on the System

1. Ensure all batteries for the Titan systems are fully charged and disconnected.
2. Turn on all ANT+ components and have them in pairing mode following the steps outlined in their device specifications, while in proximity with Titan.
3. Wait for 5 seconds once all ANT+ devices are on.
4. **Connect the battery for the (main) front system** (labelled “FRONT MAIN”).
5. Wait until the display turns on and displays the Raspberry Pi boot screen. This should take no longer than 10 seconds after the battery is connected.
6. **Connect the battery for the rear system** (labelled “REAR”).
7. Wait until the display turns on and displays the Raspberry Pi boot screen. This should take no longer than 10 seconds after the battery is connected.
8. **Connect the battery for the spare (front) system** (labelled “FRONT SPARE”).

How to Configure/Reprogram the Raspberry Pi(s)

1. Connect a keyboard and mouse to an available RPi USB port
2. Connect a full battery to the system of interest
   a. Note that if a system test is desired, follow the steps in the previous section
3. Allow the RPi to boot and launch the Titan code as normal
4. Once the code is running press ***Ctrl+Alt+T***, this will open a terminal window that is focused.
5. Type “***sudo pkill python***” and press enter. This will end the Titan code once entered successfully.
6. If it exits to command line interface (default on boot) type “***startx***” into the terminal and hit enter to launch the graphical user interface.
7. Edit things as you intend to.

How to Reprogram the Microcontroller

Please refer to here for a more in depth explanation and how to set it up:
https://github.com/stm32duino/wiki/wiki

1. Open the Arduino IDE
2. Connect the microcontroller using a MicroUSB cable

3. Type the code
4. Configure the Arduino IDE to upload to the right serial port, using the right settings:
    a. Board: **Generic STM32F103C Series**
    b. Variant: **STM32F103CB**
    c. Upload Method: **STM32duino Bootloader**
    d. Clock Speed (MHz): **72**
5. Upload the code to the microcontroller

System Overview

The Titan electronics system is essentially 3 individually powered and configured RPis with RPi cameras that communicate data between one another using an STM32 microcontroller as a central router that also collects some data (i.e. battery levels).

Raspberry Pis (RPis)

These microcomputers are primarily meant to display the video feed from the cameras to the riders. Their secondary function is to collect/provide/display information to the rider/team. The code they run is entirely in Python 2.7 (due to the library used for overlays being based in 2.7). All this code is in the files within "***Desktop\TitanVisionFirmware\src***", these are meant to have descriptive names and be thoughly commented. An overview of the Python files in rough order of relevance:

- "bikePi.py" – This is the code that runs on the main displays (ones with data), it calls to most of the other files in the folder.
- "sparePi.py" – This is the code that runs on the spare display(s), it simply displays the camera feed and does nothing else. This reduces the possibility of error in the system.
- "config.py" – Stores several variables that are used to configure the behavior of an individual RPi. Note that these settings are only reflected on the Rpi with the file!
- "boot.py" – This is the file run on boot and is solely used to start either the basic (sparePi.py) script or the complex (bikePi.py) script based on a variable in "config.py".
- "Osd2019.py" – Used to operate the on-screen display (OSD) on the main systems
- "racesim.py" – This file contains the code required to run the internal Titan simulation to generate a 'performance factor'.
- "racesim_config.py" – Used to configure the aforementioned "racesim.py".
- "ant_module.py" – Used to setup and operate the ANT+ devices (e.g. heart rate monitors)
- "serial_coms.py" – Used to setup and test the serial communication between RPis and the microcontroller.

NOTE: ANY AND ALL CHANGES TO THESE FILES IS LOCAL TO ONLY THAT RPI! Changes need to be copied on all other systems to be implemented across all RPis. The USB drive should assist in this.

Microcontroller

The microcontroller we use is based off an STM32