Overview

We will be using 2 RasPi's for this project.

1. **TrainBot01** – This is the RasPi that is mounted on the train. The train itself will get power from the tracks. The RasPi is currently getting power from a battery bank as it must not lose power if the train comes off the rails. I have a spare MoPi that I will add and set it up to get power from the rails and some backup batteries. That way we can cleanly start and stop it and also not worry if it loses contact with the rails and we will not need to worry about the power source running out. This will connect to the other RasPi that is set up as an AP.
2. **Trainbot02** – This is the RasPi that will do all the processing. It will need to be set up as a Wi-Fi Access point (TrainBot). It will also need OpenCV and be set up to connect to a WiiMote. It will have display that shows the live video feed and the processed video feed so we know what it is seeing. There will also be two text fields that will display the output from the proximity sensor and also status comments from the train, so we know what it thinks it is doing. There will be a slider that will allow us to drive the train from the screen. Lastly it will have the ability to select a mode for the train. The modes will be Manual (controlled by the on screen slider, Manual (controlled by the WiiMote) and Auto Drive (the train starts to move forward at a speed controlled by the slider above. It will be "looking" for obstructions, signal lights and some form of Barcode or QR code to tell it what to do).

Tasks

We will need to write:

1. Basic control program for TrainBot01 – This will give us simple control of the train. This will allow us to confirm that the train does actually run. Currently this is controlled by the WiiMote.
2. Server program to grab the video footage and make it available to the TrainBot02.
3. Server program to receive motion controls from TrainBot02 and action them.
4. Server program to get proximity information and make it available to TrainBot02.
5. Master control program. This will display the information we are getting from TrainBot01 work out what to do and send the commands to TrainBot01 to action.

Division of labour

We currently have a section of track made by Alex that can be set up for different scenarios (person/animal on track, signal lights, and QR codes). We also have a round track that can be used to send the train on longer Journeys. I would suggest the following:

1. Alex to manage the track and layout.
2. Cosma to write initial graphical display and make sure that the basic functions (Camera, Proximity, Drive) work on the train itself.
3. Phil to work out how to stream the video footage and also work out how to get the three servers to run on TrainBot01 simultaneously.
4. Flynn to write server for getting proximity info and make it available to be grabbed.
5. Josh and Joshua (probably with help from Flynn) to work out how to receive commands and send a response for the train movement server.
6. Flynn, if you can see if you can write an interface similar to the one I am working on that is on a web page on TrainBot02. That way any PC connecting to the AP will be able to control the train. This is low priority and should only be attempted after other tasks are done.

Additional Functions

Once the functions listed above are working we can see what useful information can be sent to the Train track side QR codes and what we can make it do. Tasks such as:

1. Stop when it sees a Red light.
2. Slow down on Amber.
3. Go when light changes to Green.
4. Wait when it sees a QR code indicating that Points are in the wrong direction.
5. Stop in correct place so that passengers can get on/off.

And anything else we can think of.

Coding Comments

We will be coding in Python 3 with the latest version of OpenCV that I can get to compile. It may be simpler to use one of the pre compiled versions of OpenCV so as not to waste time. I do not think we are using any advanced ‘new’ features in the latest versions of OpenCV.

The drive module is called drive.py. It is quite simple as it only needs to stop/start and control the speed of one motor. Flynn, I suggest you write your own version, call it driveFlynn.py, that has the correct commands for your RasPi and motor controller. We can swap them about as necessary.

I have some proximity code that I can provide to you to make that task simpler.