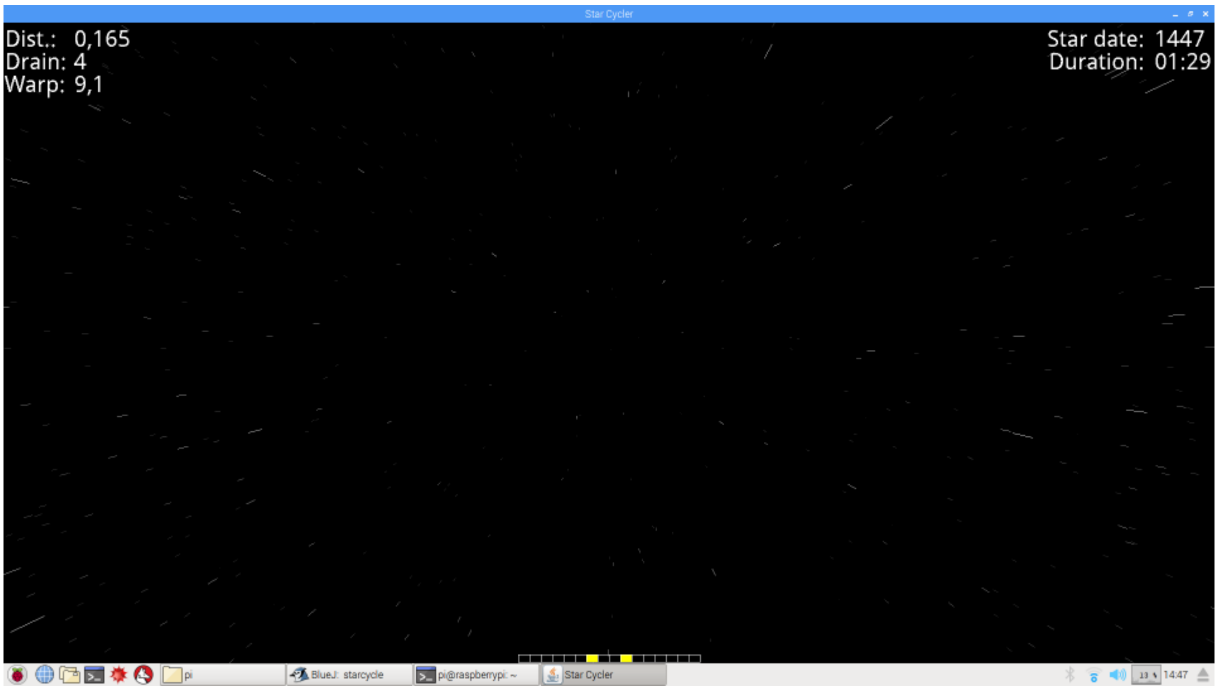
A short introduction to the project “Star-Cycle”

Introduction

Star-Cycle is a project that probably is of minor use in every-day life but surely has a high nerd-factor. The idea is to connect an exercise-bike to a Raspberry Pi, so that the Raspberry Pi can simulate the main screen of the Enterprise-bridge. The cyclist then is the warp drive and determines the speed he is cycling through the stars. On screen, some information regarding his performance is displayed: The speed on which he is going (fictive warp-factor), the distance he went (in km), an estimation for the energy he has used (in calories), the current date and time (in a somewhat strange stare-date format) and the time he is cycling.


```
Dist.:  0,165
Drain: 4
Warp: 9,1
```
```
Star date: 1447
Duration: 01:29
```

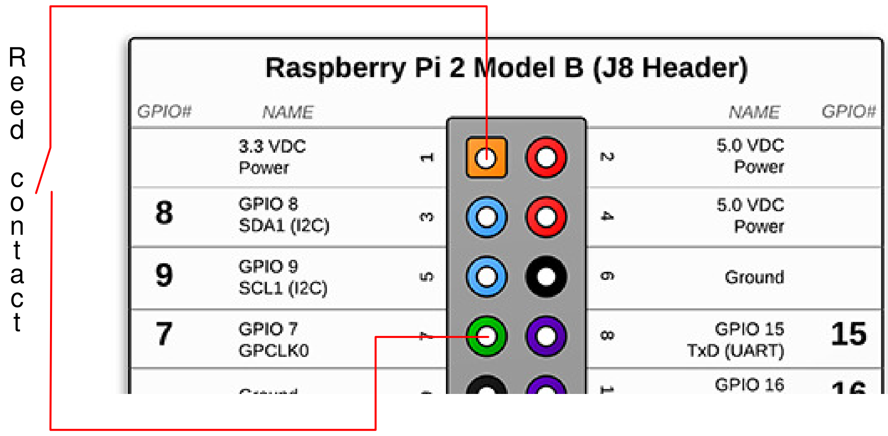
The program is written in Java, using the BlueJ-environment provided on the Raspberry Pi. It was developed on (and for) model 2B, but since one of its cores is more than sufficient even using high display resolution, it probably also will run on the older models.

Only very limited hardware skills are needed and it should be possible to use nearly every exercise-bike model.

Hardware

The sole signal needed by star-cycle is a pulse on one of the Raspi's IO-ports on every revolution of the pedal levers. My exercise bike has a tachometer generator, consisting of one reed-switch that is closed once per revolution. Normally, this switch is connected to the exercise-bike’s board-computer (via a mini TS-plug). For use with star-cycle, it is connected to the 3.3V pin and GPIO 7 on the Raspi's pin-head.

If your exercise-bike doesn't provide such a switch (or you don't want to modify it's circuits), it should be easy to add an external switch to the exercise bike: Simply glue a reed-switch somewhere near one pedal lever and stick a magnet to the lever so that the lever closes the switch once every revolution. The switch is then connected to the 3.3V signal and the GPIO 7 signal on the pin-head (see picture and http://pi4j.com/pins/model-2b-rev1.html).



Of course, it is possible to use every other GPIO pin by applying simple adjustments to the program (see below).

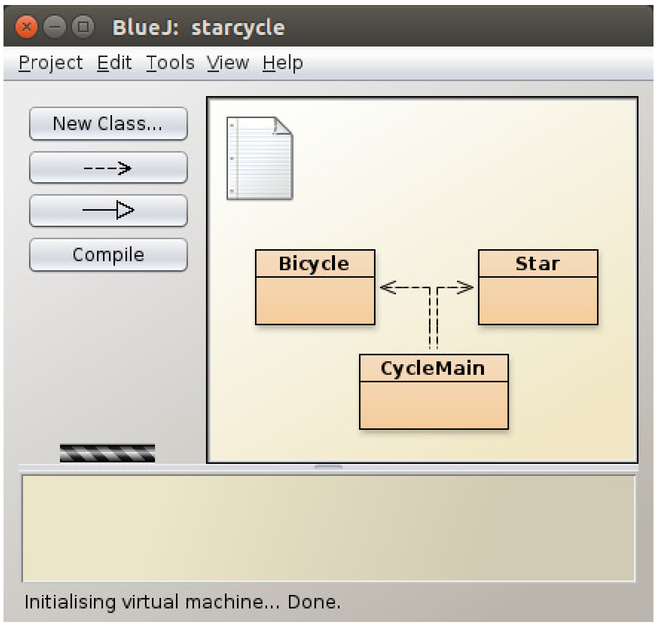
WARNING: Do not connect your exercise-bike's circuitry directly to the Raspi unless you know exactly what you are doing, since both devices can be damaged that way.

Software

The software is provided as a BlueJ project. The BlueJ-development environment, including the library necessary for using the Raspi's GPIO pins (namely pi4j), is already installed on recent Raspian versions.

Opening the BlueJ-project

Put the "starcycle"-folder somewhere in your workspace. Open BlueJ via the main menu (category "development"). Then open the project via "project/open project" (and navigating to the "starcycle" folder). You should then see the following picture (the class diagram of the program):



Initialising virtual machine... Done.

(For a more in-depth introduction to BlueJ see: http://www.bluej.org/raspberrypi/)

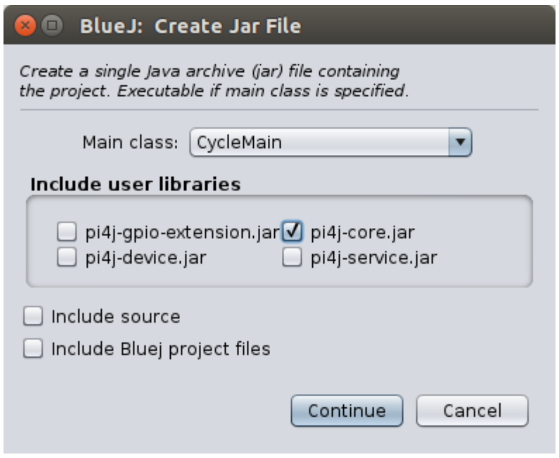
Compiling and running the program

If necessary, the program can be recompiled via “Tools/Rebuild package”.

To start the program, right-click on “CycleMain” and select the line “void main(String args[])”. Simply acknowledge the next pop-up (“BlueJ method call”) with “OK”.

If you want to start the pre-compiled program without using the BlueJ-environment, this can be achieved by entering "sudo java -jar starcycle.jar" into a console-window (pi4j needs root-rights).

To create a new version of the jar-file, choose “Project/Create jar File …” (after compiling). In the following dialog choose the following options:



BlueJ: Create Jar File

Create a single Java archive (jar) file containing the project. Executable if main class is specified.

Main class: CycleMain

Include user libraries

- [ ] pi4j-gpio-extension.jar
- [✓] pi4j-core.jar
- [ ] pi4j-device.jar
- [ ] pi4j-service.jar

- [ ] Include source
- [ ] Include Bluej project files

Closing the program

The program can be closed as usual by clicking on the “x” symbol in the title-bar.

Modifying the program

By clicking on one of the three main classes, the source of that class will be opened in an editor. It can be modified and compiled there. If you modify one of the two aggregated classes (“Bicycle” or “Star”), you have to compile the main class as well. All necessary compilation can be done by one click on “compile” in the class-view (that was opened initially).

The “Bicycle” class implements the blueprint for a bicycle-object (generated by the main program). It is used for interfacing to the exercise bike. The event-handler method (“handleGpioPinDigitalStateChangeEvent()”, implicitly declared in the constructor) just measures the time delta between two pedal-pulses and accumulates the used energy (a fixed estimation per revolution). This handler also determines, which GPIO pin should be used.

The getter methods of this class compute the other parameters (speed, distance …). For these computations a couple of constants are declared at the beginning of this class. You probably want to experiment with these constants: DRAIN_FACTOR (the calories used per pedal revolution), REV_DIST (the distance the bicycle runs on one pedal revolution (in km)), RETARD (the retard factor for the bicycles speed without cycling) and MIN_SPEED (slowest possible speed before bicycle tilts).

The Star-class mainly implements the position of a star and the methods to plot it on the screen. A star can be moved (in fact, the stars are moved in this program, not the viewpoint, i.e. the cyclist, which is always positions in the center (0, 0, 0)), drawn, and re-positioned (i.e. positioned in the background if it becomes invisible). Furthermore each star has a pointer (next) used to link stars in the order they are set relatively to the viewpoint, farthest first. The methods to achieve that are move(), draw(), rePostion(), getNext() and setNext() respectively.

The class that brings the bicycle and the stars together is CycleMain. The main() method instantiates the window, creating one object of the subclass "StarPanel" that does the main work. The 50 ms hart-beat (resulting in 20 pictures per second) of this class is created by the thread "repainter" that causes a repaint of the StarPanel-object. Repainting is done in the call-back method "paintComponent()" that sets the background to black, moves and draws all stars and eventually prints the information fetched from the Bicycle object on the foreground. Drawing of the stars is done beginning with the ones that are farthest away from the viewpoint to allow over-painting of stars in the background by those that are more in the foreground.

The position-values for the bicycle related information on the screen are results of some experimenting. If you like, you can alter them by modifying the parameters of the "textOut()"-calls.

Future enhancements

- After some time, the more or less uniform display gets a little bit boring. This could be overcome (at least partly) by showing different objects from time to time. Implementation could be done by drawing pictures onto the screen instead of points and lines, e.g. star-ships, planets, galaxies …
- Another possibility would be to introduce random movements of the viewpoint, e.g. translations, direction-changes or rolls.
- It might be interesting to calculate some statistical values on your journeys and store them for later evaluation, probably separately for different persons.
- If the program also could read the training level selected or even could alter it, it is possible to calculate a better approximation of the calorie drain or even to simulate a course with some slopes.

Finally

Have fun implementing your star-cycle, so that you are able in the near future

… to boldly cycle where no one has cycled before :-)