Computer Science Block 1: The Traffic Light Challenge

October 2015

1 The Challenge...

You are part of a team working as consultants for Transport for London on investigating the use of technology to help make improvements to the road network.

The challenge your team are faced with is to design and prototype a traffic light system to safely control the flow of traffic along a road. As an example, in the road layout shown in figure 1, roadworks along the main road mean that a temporary traffic light system must be installed, with two sets of lights located at the two points marked Z, allowing traffic to flow alternately from left-to-right and then right-to-left.

You can assume that the distance between the two sets of lights is 200m and that the speed limit along this stretch of road is 30mph.

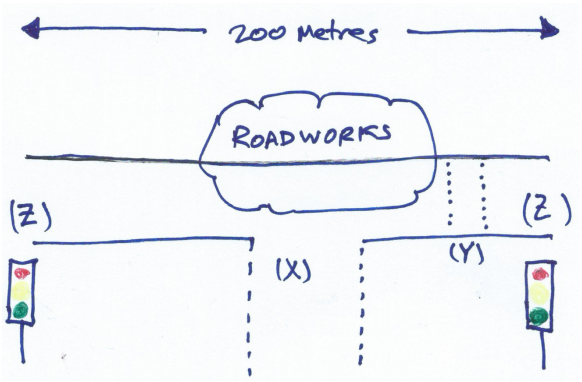

Figure 1: A Road Layout

As this work is exploratory in nature, the client is interested in innovation and designs that creative responses to the problem, rather than the delivery of complete working products . So that means that an interesting idea or an unusual experiment may be just as good as a more functional but less innovative prototype.

To make things more difficult, the Chief Designer in your organisation has resigned and departed

in rather a hurry. Before leaving, she had begun working on this problem and had come up with some initial ideas. Fortunately, some of her early work, in the form on notes, prototypes and even some initial code, was left behind, and you can use this as a starting point - either as something to build upon and extend, or as something to depart from.

2 The Task

The task of designing and prototyping the traffic light system will probably involve several elements, possibly including:

Represent the behaviour of your system in an appropriate and useful way, so as to help you think about the problem, communicate your ideas to others and to record ideas you have had in a way that you will be able to understand at a later date. An obvious idea is to represent the system as a Finite State Machine, in the form of a State Transition Diagram. This will identify the different states that the system can be in, as well as transitions between these states, and can provide you with a tool to think about major aspects of the design.

Construct the hardware including the two sets of traffic traffic lights. One way of doing this is to represent traffic lights as LEDs connected via an Arduino to the computer running Racket.

Prototype the design's behaviour, e.g., by implementing the FSM as a Racket program that will control the lights connected to the Arduino.

Conduct a systematic test of your design to ensure that it functions correctly.

Identify limitations and issues with your solution.

Document your design either so that someone else (e.g. another developer joining the team later) could understand what the design is and how it works, or to persuade someone (e.g. the client) that this approach is a good idea. The medium for this 'documentation' could be more or less anything (a brief written report, a short video, song, etc) that helps to convey the message. The important thing is that the 'documentation' should do an effective job of representing your design for the intended audience.

As part of implementing the prototype, there will be several important decisions to make about how the desired behaviour will be achieved, for example, how state transitions will be triggered. One possible option here is to provide a button, connected via the Arduino, that when pressed, will trigger a transition from one state to the next. A more sophisticated approach will use a timer, and transitions will occur after a specified time has elapsed. Clearly different times are appropriate for different transitions (e.g. a traffic light will typically remain green for longer than it is amber).

3 Resources

To complete the project you will have the following available:

- Arduino (running Firmata), LEDs (red, orange, green), resistors, push buttons, any sensors you may need, and so on.
- Racket, the `firmata.rkt` library.
- Sample racket code showing a simple prototype for a single traffic light.

4 Starting Point

When the former Chief Designer left the organisation, her desk was searched, and various ideas, examples and prototypes for this project were discovered. You can investigate these materials as a starting point.

In the prototypes created by the former Chief Designer, the lights themselves are coloured LEDs connected to an Arduino board (with appropriate resistors). The Arduino itself runs the ASIP sketch - so it is capable of receiving ASIP[1] commands from a host computer connectedly USB, and setting output pin value (to which LEDs etc can be connected) and of reading input pin values, which are then sent back to the computer. The prototypes work by having Racket program running on the host computer

[1] Check!

An example is provided that implements a single traffic light.

An important observation is that a traffic light system can be described as a Finite State Machine. The system moves from one state to another upon the occurrence of events, which could be the passage of a specified amount of time, a human action such as pushing a button, or the detection by sensors of the cars arriving at one or other of the traffic lights. The program `SimpleSingleTraffic.rkt` defines a simple sequence of light states, as represented in Figure 2, and cycles through this sequence, pausing for a fixed interval in each state. Figure 3 shows

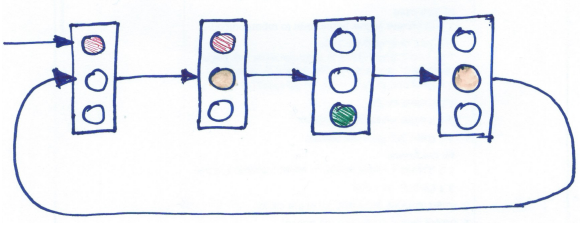

Figure 2: State sequence for a single traffic light

how this might be constructed on a breadboard. As this might be rather unclear, Figures 4 and 5 show a clearer layout as well as a schematic diagram. Note that the Green, Orange and Red LEDs are connected to pins 11, 12 and 13 of the Arduino board respectively.

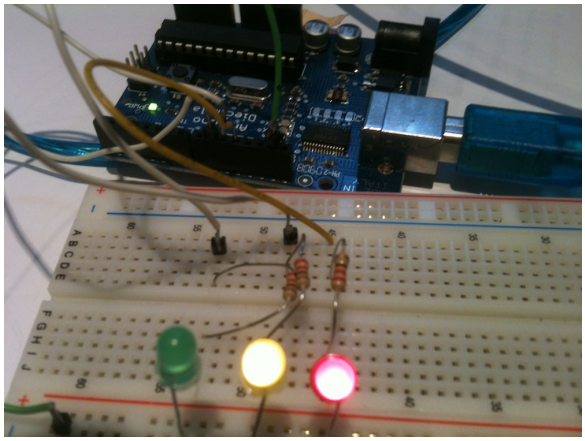

Figure 3: Arduino-based implementation of a single traffic light

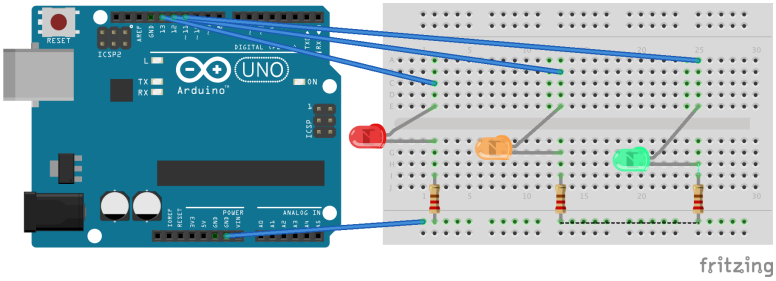

Figure 4: Arduino-based implementation of a single traffic light

Another Racket program was found among the materials left by the departed Chief Designer. It appears to implement two sets of lights, as we;ll as implementing FSMs in a more general way, permitting arbitrary state transitions - not only a linear sequence of states. `DoubleTraffic.rkt` appears to use a tabular representation of the the FSM. No hardware design has been found that corresponds to either design, but both appear to assume that a push-button have been connected to pin 6 of the Arduino board. Pressing the button moves the FSM from one state to the next.

Note that using Racket and Arduino connected in this way using the ASIP protocol is an obvious (and quite sensible) way of approaching the problem. However there are many other ways of prototyping your solution.

5 Extensions

To further develop your understanding you can extend the challenge with one or more of the following:

- Add a pedestrian crossing at point (Y) with red and green lights for pedestrians. The lights

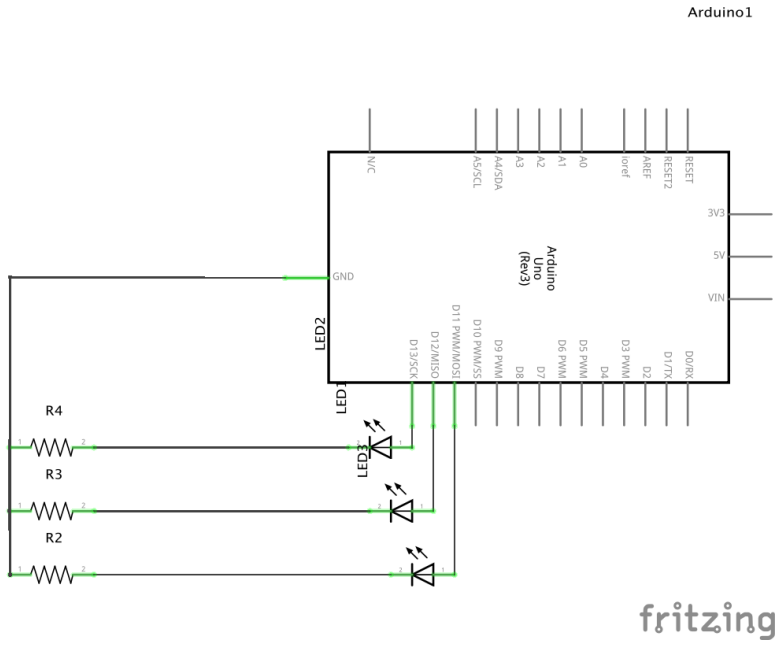

Figure 5: Schematic diagram for a single traffic light

of the crossing must be synchronised with those of the main traffic lights.

- Add flashing lights - for instance, the green flashing indicator for pedestrians to show that it is safe to complete a crossing, but not to begin to cross.
- Add additional traffic light at the intersection (X), where a minor road joins the main road.
- Sensors to detect the presence or arrival of cars waiting at the points (Z) so that if no cars are present in one direction, those travelling in the opposite direction will continue to have a green light.

As the client is interested in innovative thinking, exploring other, more experimental ideas will be welcomed.

6 Arduino and Racket

The red, orange and green lights could be implemented as LEDs connected to an Arduino board. The Arduino can in turn be controlled by a Racket program running on a computer, that communicates via a USB cable with the Arduino, sending commands and receiving data.

This approach will connect a Racket program running on a regular computer to an Arduino that will, in turn be connected to LEDs, switches, and so on. A library called `ASIPMain.rkt`. implements a protocol known as ASIP, which provides a language that allows computers to communicate with a microcontroller such as the Arduino. The ASIP protocol provides a mechanism

for the computer to control the Arduino by setting and clearing output pins, detecting the status of input pins, and so on.

To use `ASIPMain.rkt`, make a copy of this file and place it in the same folder as your Racket program. Functions provided by the `ASIPMain.rkt` package that are available in a Racket program include:

Setup	
`(open-asip)`	Creates a connection to a connected Arduino.
`(close-asip)`	Close the connection to the Arduino.
`(set-pin-mode p m)`	Set the mode of pin `p` to mode `m`, which can be `INPUT_MODE`, `OUTPUT_MODE`, `INPUT_PULLUP_MODE`, `ANALOGUE_MODE`, `PWM_MODE`. This must be done before other commands are issued to tell software running on the Arduino how the pin is going to be used.
Output	
`(digital-write p v)`	Set pin `p` to value `v`.
Input	
`(digital-read p)`	Digital value of pin number `p` (0 or 1)
`(analog-read p)`	Return the value of an analogue input pin.

In order to use the ASIP protocol and control an Arduino board from Racket, the following important steps are necessary:

- Run the ASIP program on the Arduino board.
- Quit the Arduino program.
- At the start of your Racket program you will need the following lines. The first loads the ASIP library, and the second opens a connection to a conected Arduino board.

```
(require "AsipMain.rkt")

(define setup (lambda  ()
                (open-asip)
                (set-pin-mode 12 OUTPUT_MODE)
                (set-pin-mode 5 INPUT_MODE)
        )
)

(define mainLoop (lambda  ()
                ;  read values from input pins?
                ;  change state?
                ;  write output pins?
                (mainLoop)
        )
)
```

```
(setup)
(mainLoop)
```