TRAFFIC LIGHT SYSTEM

Embedded Systems Project

Ciara Power
20072488
BSc in Applied Computing

Introduction

For my embedded systems project, I developed a traffic lights system using the XMC 2Go microchip. The basic elements of the design include components from an Arduino Uno kit also.

Traffic lights are a necessity in today's world, and are seen on every major street from one corner of the globe to the other. For this reason, I thought it would be interesting to try replicate the basic functions of a traffic light system, with a pedestrian button feature.

Of course, in a real world situation, traffic lights have many more sensors and functionalities than I will be exploring in this project, such as possible weight detection for cars in certain lanes, or even cameras with integrated car detection functions.

System Description

There are three LEDs used as the traffic lights – red, yellow, and green. These are set up on a breadboard and connected to pins on the XMC 2Go, configured as Digital Input/Outputs on the Dave app. Initially, the green light will be on and the other lights will be off.

A pushbutton is used to act as the pedestrian button. This is connected onto the breadboard, and to a pin on the microchip, and set up as a Dave Input/Output app also.

As in the real life pedestrian crossings, I tried include a buzzer in the system design. This was also connected on the breadboard, and to a microchip pin, and set up as an Input/Output in the Dave app. However it did not work (the buzzer only blipped when set to high , instead of buzzing for 5 seconds) , so I removed it from the design.

The Dave application will include multiple timer apps, used to time each period between traffic light changes, and to debounce the button input.

Functionality

These traffic lights will have a somewhat basic functionality.

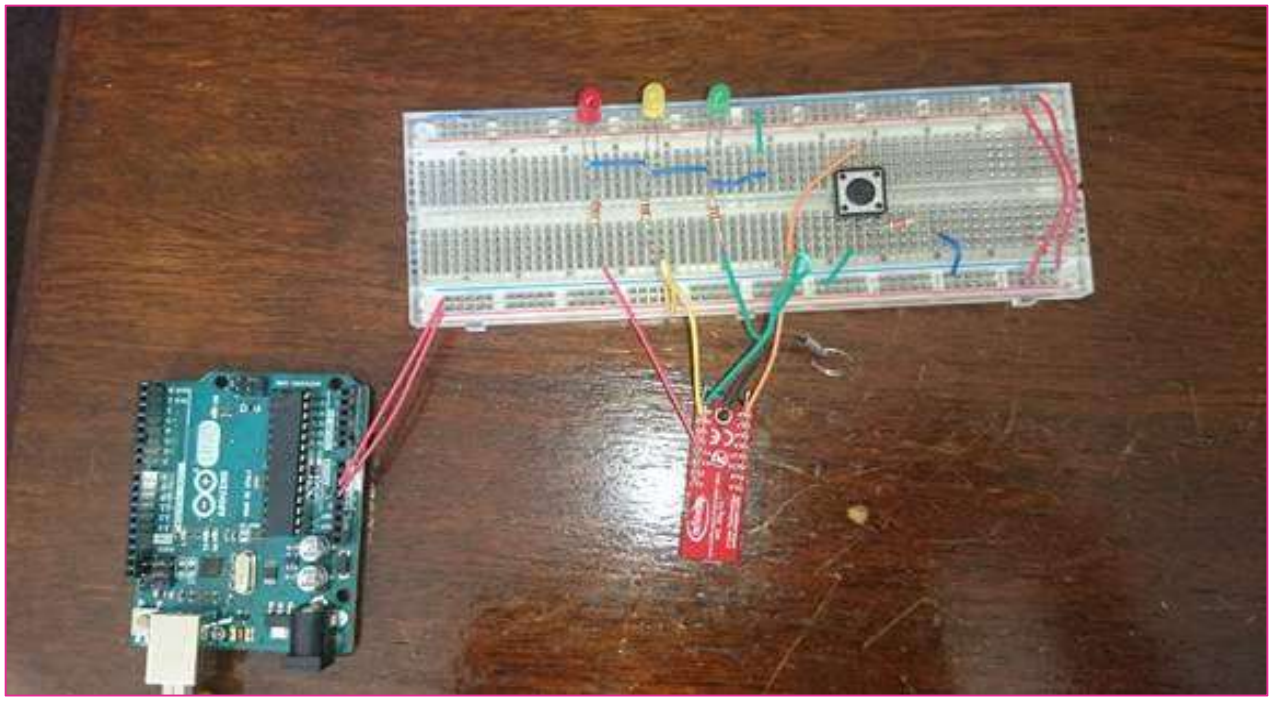
Initially, the green LED is on, and the yellow/red LEDs are off. The buzzer is silent at this stage (if included). The button is initially unpressed.

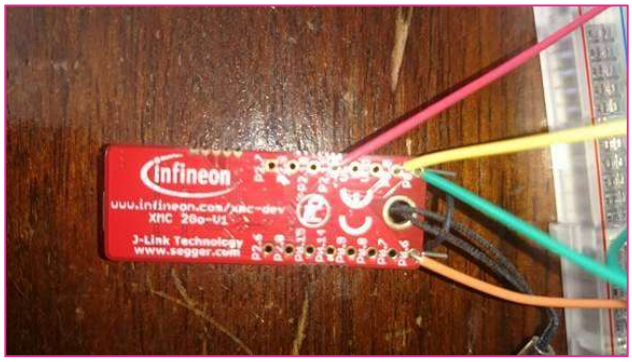
The lights will stay in this position until the pushbutton is pressed. When this happens, the traffic lights change. The green LED will turn off, and the yellow LED will turn on for 3 seconds. The yellow LED will then turn off, and the red LED will turn on for 5 seconds. This is the pedestrian crossing time, so from my original design, the buzzer should begin to sound for the 5 seconds the red LED is on also (to stop the cars). The green LED will then turn back on, and the traffic lights are back to the initial position.

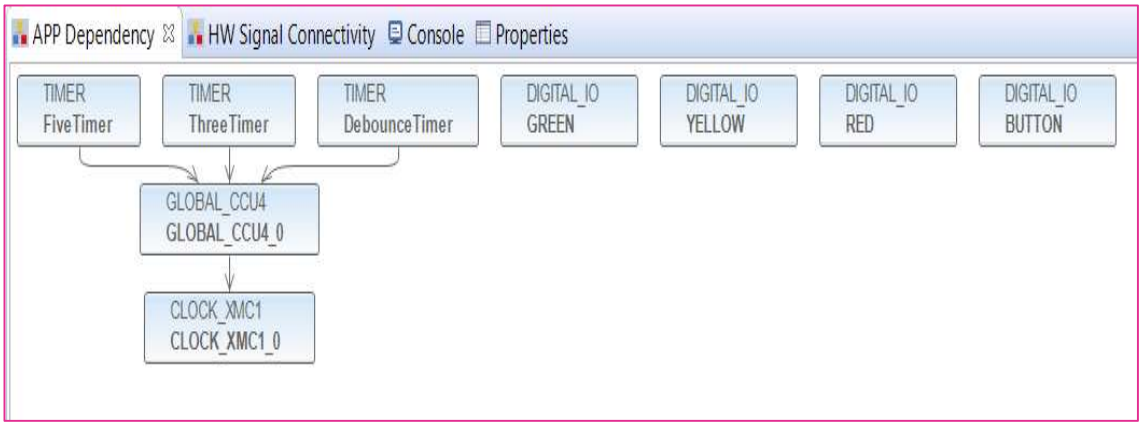
Equipment

- Arduino Uno (for power supply)
- Breadboard
- LED x3
- Pushbutton x1
- Wires
- 10kΩ Resistor x4
- XMC 2Go Microchip
- Buzzer (original design)

Setup Diagram







The apps used are shown in the diagram below.

Results

The system worked as expected, without the buzzer function added in.

A short video of the working design is included in the accompanying folder for this document, the code used is included in the Code section below, and the Dave files are also included in the project folder also.

Conclusions

I had difficulties at first attempting to set up the various hardware input and outputs, as I had never worked with components from the Arduino Kit before.

Once I had one basic LED working through the Dave code, I found adding the other LEDs and pushbutton much easier. The buzzer was not working as it should but I felt maybe it was an issue with my choice of resistor (10kΩ).

Code

```
/*
 * main.c
 *
 *  Created on: 2017 Apr 26 10:02:22
 *  Author: ciara
 */



#include <DAVE.h>                 //Declarations from DAVE Code Generation
(includes SFR declaration)

/**

 * @brief main() - Application entry point
 *
 * <b>Details of function</b><br>
 * This routine is the application entry point. It is invoked by the device
startup code. It is responsible for
 * invoking the APP initialization dispatcher routine - DAVE_Init() and
hosting the place-holder for user application
 * code.
 */

 void changeLights(){
       // green off, yellow on
       DIGITAL_IO_SetOutputLow(&GREEN);
       DIGITAL_IO_SetOutputHigh(&YELLOW);
       TIMER_Start(&ThreeTimer);
       while(!TIMER_GetInterruptStatus(&ThreeTimer));
       TIMER_Stop(&ThreeTimer);
       TIMER_Clear(&ThreeTimer);
       TIMER_ClearEvent(&ThreeTimer);

       //yellow off, red on
       DIGITAL_IO_SetOutputLow(&YELLOW);
       DIGITAL_IO_SetOutputHigh(&RED);
       TIMER_Start(&FiveTimer);
       while(!TIMER_GetInterruptStatus(&FiveTimer));
       TIMER_Stop(&FiveTimer);
       TIMER_Clear(&FiveTimer);
       TIMER_ClearEvent(&FiveTimer);

        // red off, green on
       DIGITAL_IO_SetOutputLow(&RED);
       DIGITAL_IO_SetOutputHigh(&GREEN);
       TIMER_Start(&ThreeTimer);
       while(!TIMER_GetInterruptStatus(&ThreeTimer));
       TIMER_Stop(&ThreeTimer);
       TIMER_Clear(&ThreeTimer);
       TIMER_ClearEvent(&ThreeTimer);

 }
int main(void)
{
  DAVE_STATUS_t status;
```

```
  status = DAVE_Init();           /* Initialization of DAVE APPs  */

  if(status != DAVE_STATUS_SUCCESS)
  {
    /* Placeholder for error handler code. The while loop below can be
replaced with an user error handler. */
    XMC_DEBUG("DAVE APPs initialization failed\n");

    while(1U)
    {

    }
  }

  /* Placeholder for user application code. The while loop below can be
replaced with user application code. */
  while(1U)
  {
       if(DIGITAL_IO_GetInput(&BUTTON)==1){
       TIMER_Start(&DebounceTimer);
       while(!TIMER_GetInterruptStatus(&DebounceTimer));
       TIMER_Stop(&DebounceTimer);
       TIMER_Clear(&DebounceTimer);
       TIMER_ClearEvent(&DebounceTimer);
             if(DIGITAL_IO_GetInput(&BUTTON)==1){
            changeLights();
             }
       }
  }
}
```