

Electronic Design 314

Gate Controller

13 May 2016

Tristan Nel (18179460)

&

Daniel Robinson (18361137)

Table of Contents

[Declaration of own work](#)

[Summary](#)

[Introduction](#)

[System Description](#)

[2.1 Microcontroller](#)

[2.2 Power supply](#)

[2.3 LCD](#)

[2.4 External UART](#)

[2.5 Motor](#)

[2.6 Infrared](#)

[Transmitter](#)

[Receiver](#)

[2.7 Other Peripherals](#)

[Design Detail](#)

[3.1 Microcontroller](#)

[3.2 Power supply](#)

[3.3 LCD](#)

[3.4 External UART](#)

[3.5 Motor](#)

[3.6 Infrared](#)

[Transmitter](#)

[Receiver](#)

[3.7 Other peripherals](#)

[Programs](#)

[Gate Controller:](#)

[global.c](#)

[adc.c](#)

[infrared.c](#)

[lcd.c](#)

[Remote Controller](#)

[global.c](#)

[Measurements and results](#)

[Conclusion](#)

[Appendix](#)

[User Manual](#)

[Connect the Gate Assembly to the Controller Board](#)

[Power the Gate Controller Board and Remote Controller](#)

[Normal Mode vs Test Mode](#)

[Normal Mode](#)

[Remote Control](#)

[Board](#)

[Test Mode](#)

[Board](#)

[PC Debug Software and UART connection](#)

[Technical Specification](#)

[Pinout](#)

[G14 Gate controller](#)

[CN2](#)

[CN3](#)

[G12 Infra-red transmitter](#)

[Circuit Diagrams](#)

[Electronic submission:](#)

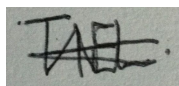
[Symbols and Abbreviations](#)

[Bibliography](#)

Declaration of own work

I (we), the undersigned, hereby declare that this report and the work contained therein is our own original work, except where indicated.

Signature

A handwritten signature in black ink, appearing to be 'TAE' with a horizontal line through it.

Date

10 May 2016

Signature

A handwritten signature in black ink, appearing to be 'D Robinson'.

Date

10 May 2016

Summary

This was an educational project for students to create a prototype gate controller board and infrared remote. It depicts the design process and shortfalls in the prototype design, and what changes can be made in the final product.

The controller board was built on a prototype PCB developed by the University of Stellenbosch Faculty of Engineering (E&E). The provided routing is minimal, the goal being that students develop their own wiring/designs.

The gate is a small-scale version of a real gate. It has proximity switches to stop the gate from moving at either end. It will deal with mechanical obstructions to the gate, for safety purposes.

It has an LCD which shows event data logs, amongst other data.

One can also control the gate using the infrared remote.

In future, it would be beneficial to add a card reader, since NFC cards are cheaper than remotes, and to add GSM/2G to the project thereby letting a cellphone app/sms/call to open the gate, which is even cheaper to implement for multiple users.

The system works as described.

If turned into a full-scale project, a proper PCB would need to be designed, routed and manufactured. A higher rated motor and motor controller would be needed. It would be easy to implement, since one would merely set the maximum PWM duty cycle percentage in the code, to account for a different motor. A housing for the remote and gate controller would also need to be developed. ESD protection and waterproofing would also be advised against lightning, rain, floods and other environmental effects. The final solution would also need better tamper-proof protection. The testing app merely sends one standard byte to set the system in testing mode, and then any individual can open the gate. One could also read the latest remote address and data, thereby creating a clone remote, if one has access to the board in a panel. It would be better to have a password generated from a checksum unique to the serial number of the specific gate.

A battery would be ideal to let the gate work for at least 2 hours, especially in South Africa, where load shedding lasts for 2 hours on average.

1. Introduction

This report details the operation and technical specifications of a mechanical gate assembly operated by a controller board and infrared remote control.

The aim of the project was to implement a functioning remote-controlled gate, with additional features such as PC demo/debug software, buzzer tones and current sensing.

Technical schematics and flow diagrams are provided, as well as the calculations for the component choices.

2. System Description

The controller board was built on a prototype PCB developed by the University of Stellenbosch Faculty of Engineering (E&E). The provided routing was minimal, the goal being that students develop their own wiring/designs.

The main hardware consists of a Renesas RL78/G14 R5F104LEA processor, a Powertip 16x1 LCD for user feedback, external FT230XS UART chip to communicate with an external test program at a 9600 baud rate, a DRV8801 motor driver (40kHz PWM), a TSOP348X infrared receiver, an MCP601 operational amplifier for current sensing, a QJT38 6V DC Motor geared motor, and a 2kHz buzzer for warning conditions.

There also exists an infra-red remote, operated by a Renesas RL78/G12 1026A processor, to control the gate.

The system includes current sensing in order to pick up motor strain or mechanical interference from an object in the way of the gate opening or closing. From thereafter it will automatically return to it's previous state (fully opened or closed).

For construction of the controller board and remote control, see the Circuit Diagrams (Appendix)

For use of the system, see the User Manual (Appendix)

2.1 Microcontroller

- Renesas RL78/G14 R5F104LEA processor

It does most of the processing. It interfaces with the other components through it's GPIO and interrupt pins.

2.2 Power supply

- 7805 5V Regulator.

It converts a 9 - 12V input to power most of the system's peripherals, which run at 5V.

2.3 LCD

- Powertip PC1601A 16x1 LCD (no backlight).

It provides visual feedback to the user, without requiring a computer. Typical information would include events (with timestamps), if the gate is moving, opened or closed, and the current drawn by the gate.

2.4 External UART

- External FT230XS UART chip

UART chip communicates with an external test program at a 9600 baud rate. The program is used to test the system and receive feedback such as event data-logs, current, status, infrared address and data from latest transmission, and to open or close the gate, and test the buzzer etc.

2.5 Motor

- QJT38 6V DC geared motor.

This opens and closes the gate.

- DRV8801 motor driver (40kHz PWM).

This is used to power the motor. It consists of an H-bridge which allows it to change the direction of the motor easily with digital logic. It also has an nSleep pin to let the motor controller sleep when not in use, which contributes to power saving. The motor controller requires PWM to control a motor, but it does have a 100% PWM mode.

- MCP601 operational amplifier for current sensing.

This is used for current sensing and to ultimately stop the gate if a mechanical obstruction interferes with the movement of the gate.

- Mechanical gate

The gate has momentary switches on either end which act as gate stoppers. It operates in an angular motion with a 90 degree angle.

2.6 Infrared

Transmitter

- BIRNM23C2DS IR LED

LED lets the gate controller system know, wirelessly, if it should open, close or emergency stop the gate.

- Renesas G12 1026A processor.

The G12 listens to buttons which will open/close/emergency stop the gate. It sends the required message with a RC5 Manchester encoding (889 μ s per bit) through an IR LED

Receiver

- TSOP348X infrared receiver

The receiver receives a manchester encoded message from the IR remote at a 38kHz carrier. It will relay the message to the microcontroller which will decode the message.

2.7 Other Peripherals

A PTC resettable fuse to protect the main microcontroller from surge spikes or shorts.

It is resettable since replacing a quick-blow fuse is too costly.

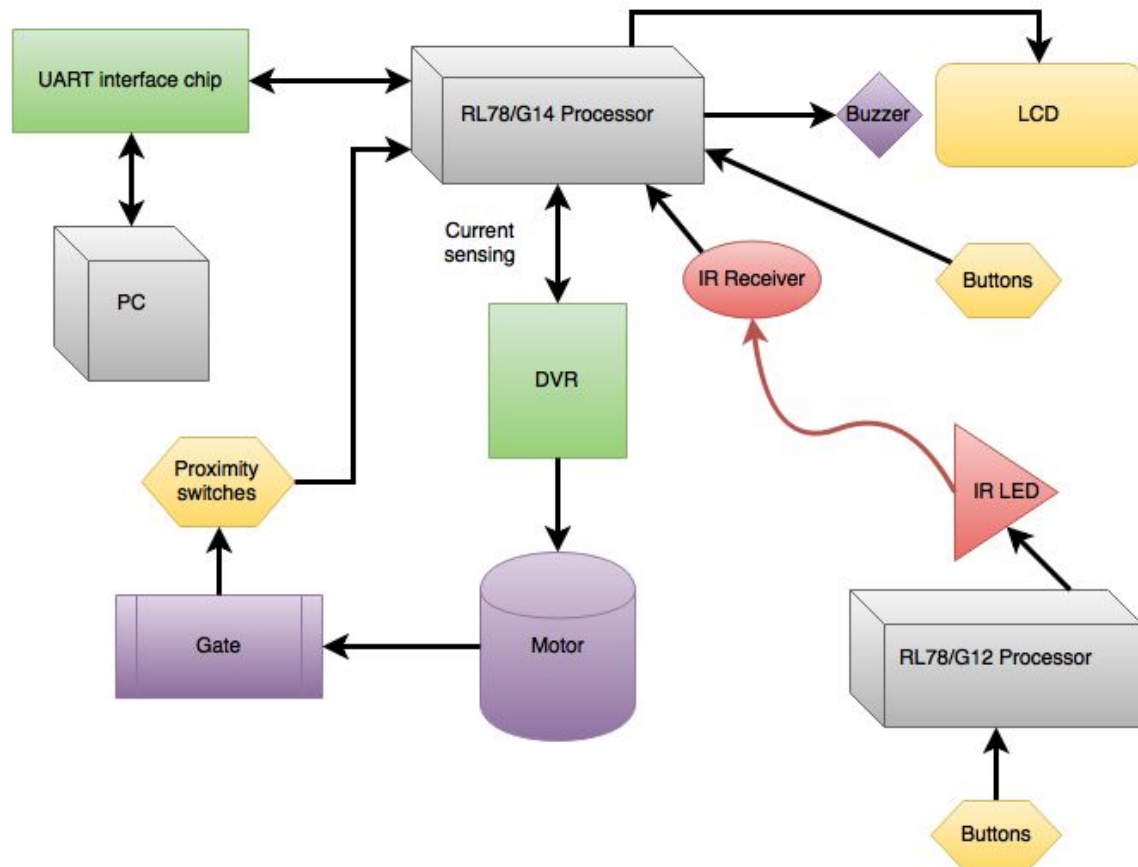
A potentiometer on the G14 promotional board which is utilized to control the motor speed.

A buzzer circuit is implemented using a 2N3904 NPN transistor, and an MT12G 2P buzzer.

An nSleep toggle switch to provide an emergency stop regardless if the processor is running or not.

A toggle switch which toggles power to the G14 promotional board. This was in an effort to have three 5V power supplies simultaneously (from PC and and 5V regulator). This enabled hardware debugging, testing and operation of the gate, without having to continually disconnect and reconnect power cables.

System Diagram



For all diagrams: Grey (processor), green (IC), yellow (user interface), purple (peripheral), red (infrared)

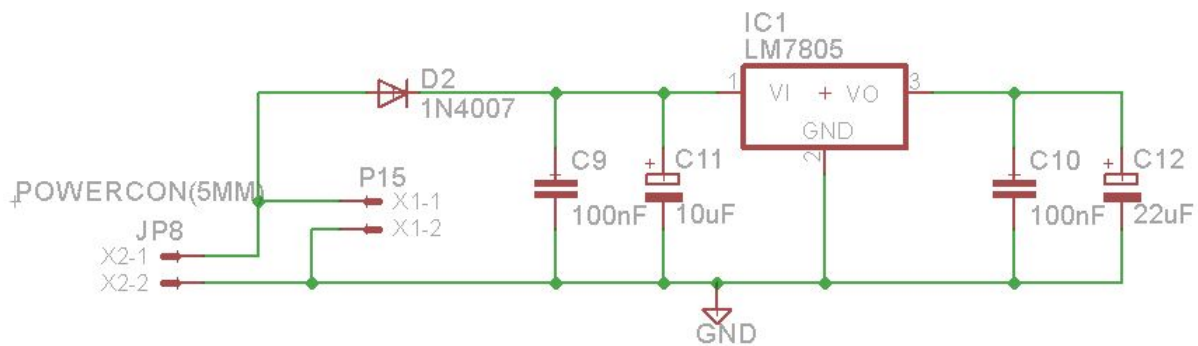
3.Design Detail

The system design process, component information and calculations are presented below.

3.1 Microcontroller

The microcontrollers used for this project are provided as per the required specification. The gate controller used the Renesas G14 R5F104LEA promotional board. The IR remote used the Renesas G12 1026A processor. As an alternative, an Arduino would work quite well since there is such a large community base to provide support.

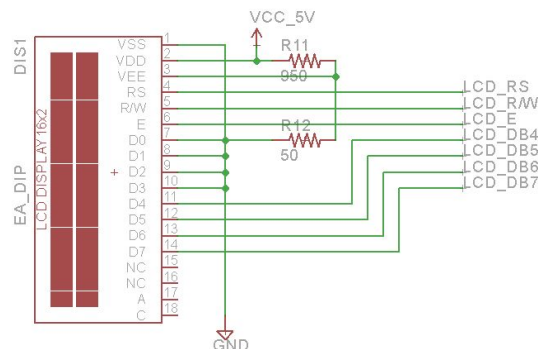
3.2 Power supply



Power supply circuit

The heart of the PSU is the 5V regulator. 5V is in sync with the specifications of most of the board, except for the motor and motor driver. Large smoothing capacitors were used to adjust for the voltage ripple in the PSU, while small capacitors were used to decrease noise and high frequency spikes. The diode prevents harm coming to components if power is connected in reverse polarity to the system.

3.3 LCD

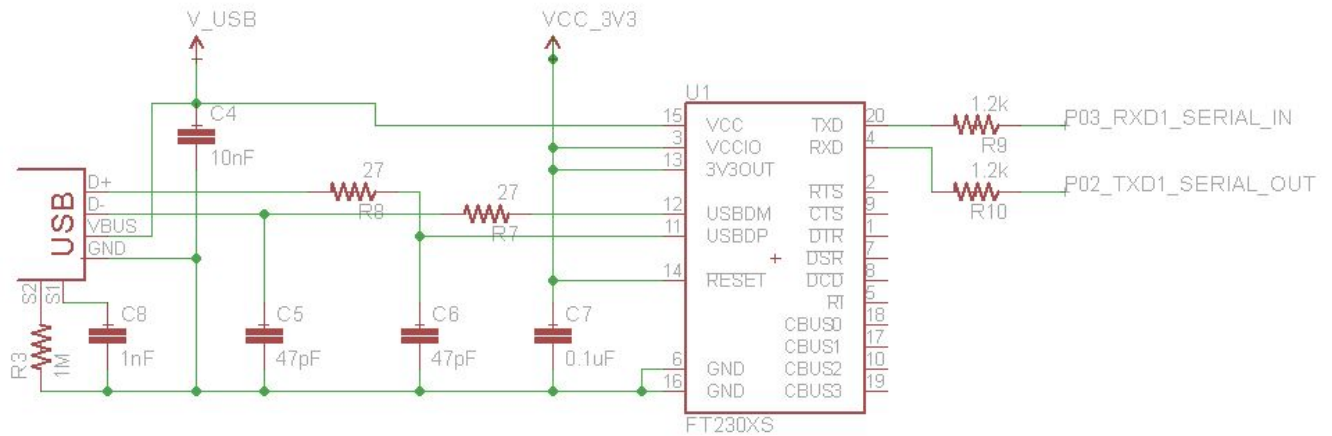


LCD module

The system uses a 16x1, 4-bit mode LCD display with no backlight. By using 4-bit mode, less connections are used. The resulting performance decrease compared to using 8-bit mode is very minimal, and not noticeable to the naked eye. Though a backlight is not used, it would be highly recommended in future, especially if testing the gate controller in a darkly lit panel. The LCD is used for feedback to the user (if opened/closed, current measurement, event logs etc). A 16x2 Display would also be recommended since more information can be displayed, and the transition would be seamless, since it fits in the same position on the prototype board. For setting contrast, a 1k potentiometer was used to find the optimum contrast setting. It is generally found that the contrast is indirectly proportional to V_{EE} ($V_{contrast}$).

3.4 External UART

An FTDI230x chip is used for serial communication as a way to test the functioning of the gate controller, using a dedicated external application.



UART circuit

The large capacitors C4 and C7 are used for smoothing/leveling, and the smaller capacitors C5 and C6 are used for filtering out high frequencies. R3C1 and R4C2 form low pass filters which filter out frequencies higher than the USB clock/data rate. The resistors R7 and R8 also limit the circuit to a smaller extent.

Tx and Rx of the FTDI chip are connected to the Rx and Tx of the Renesas board, respectfully. More specifically, UART1 is used, instead of UART0, since hardware debugging can occur on UART0 and testing through the external testing application on UART1 can occur simultaneously.

R9 and R10 have a value of 1.2kΩ, which limit the current between separate 5V power supplies. More specifically, if there is a slight discrepancy between the UART power supply from a PC and the 5V rails from the regulator, then due to

$$V = IR$$

and the low resistance between the two, a large current would flow.

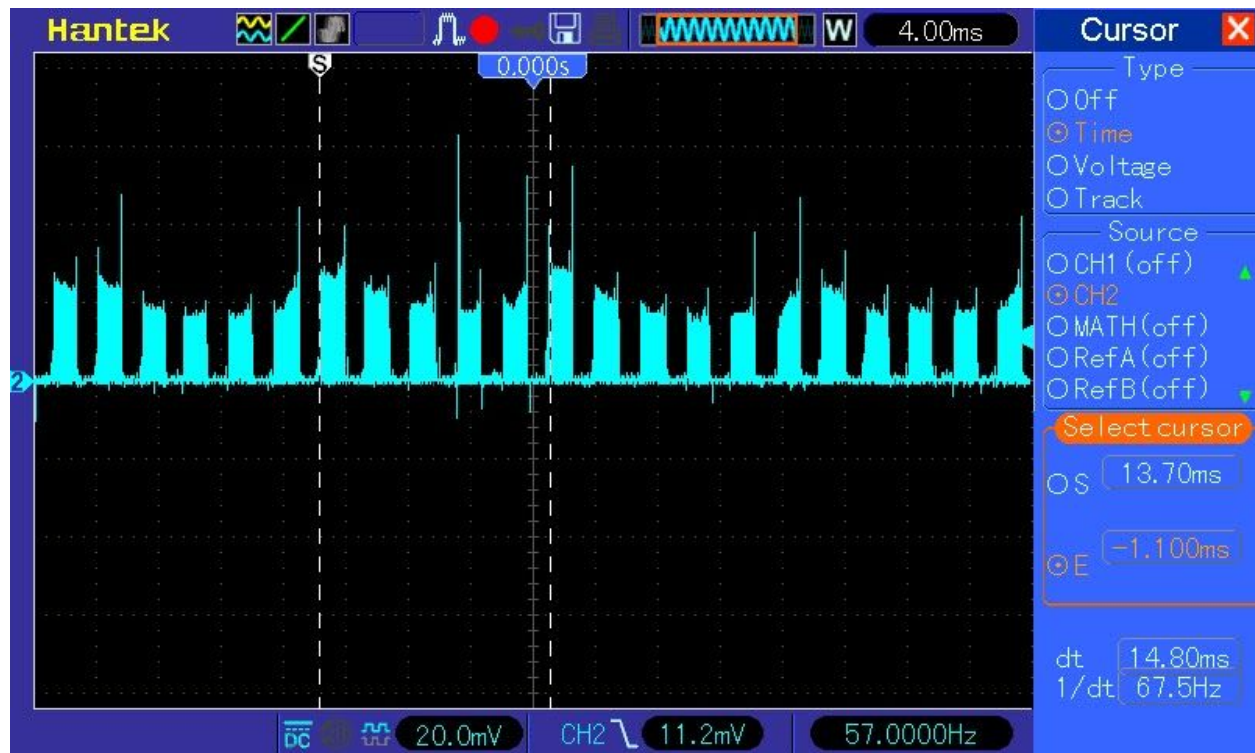
3.5 Motor

The geared motor was driven with pulse-width modulation (PWM), in an effort to set a comfortable speed for the gate controller. Also, as the motor was rated for 6V, and the motor controller was rated for 8V minimum, the only way to feed an average voltage of at most 6V to the motor was through the use of PWM.

A frequency of 40kHz for the motor was chosen since it is much higher than the natural frequency of about 150 Hz (from the datasheet) for the motor. If:

$$f_{PWM} \leq 150 [Hz]$$

then the motor would run rougher, which would quite possibly be audible as well.

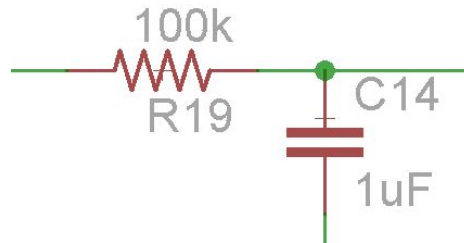


Voltage over 0.27Ω shunt resistor before filtering

It was observed that the natural frequency of the motor is actually 67.5Hz.

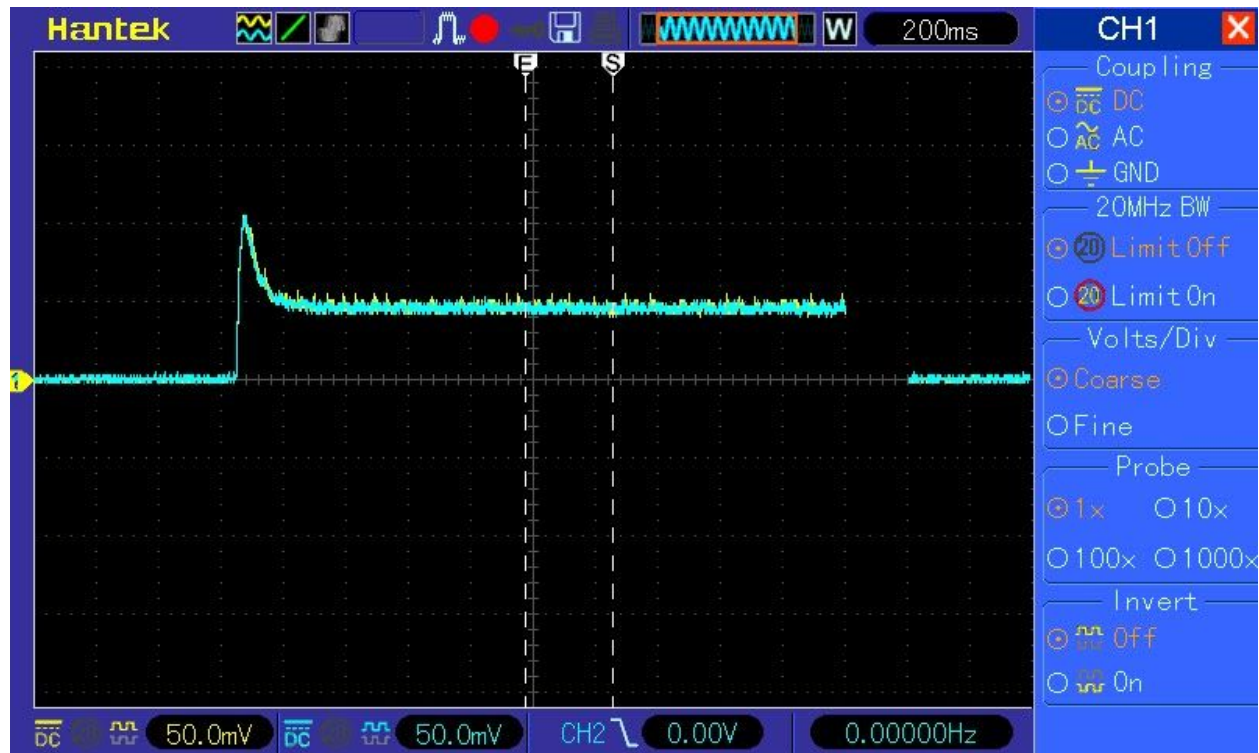
Ideally we would like a very stable value that tracks the current at a rate slower than the natural frequency. 10Hz made for an ideal time constant because it is smaller than 67.5 Hz by a factor of more than 5. It is also a speed with which, if synchronised with the rate of ADC feedback, is visible to the naked eye.

10Hz means an RC time constant of 100ms with $R = 100k\Omega$ and $C = 1\mu F$. The resistor value was chosen first so as to make sure that the measurement current is much smaller than the current being measured.



RC filter

There is also an internal differential amplifier inside the motor driver whereby we can measure the current out of a pin called V_{propi} . V_{propi} is, according to the datasheet, 5 times larger than the voltage over the shunt resistor.



V_{propi} after filtering with RC time constant of 0.1 seconds

The true current value is:

$$0.05/5/0.27 = 37[mA]$$

Here it was observed that the measurement was quite smooth. It would, however, need to be amplified, otherwise it would be too insensitive to changes in current due to motor strain. The

G14 has 10-bit ADCs, which meant that one could measure 1024 steps of accuracy from 0 - 5V. In other words, about 5mV per step. Unfortunately, the resolution is not strong enough to measure very low currents or changes. The ADC would only pick up motor strain at at least (in this case) 50 +/- 5mV.

All the current which passes through the motor driver passes through a 0.27Ω resistor, which is grounded on one end. The motor is rated at about 100mA, but usually runs at about 90mA peak, with an average of about 37mA.

This is the voltage across the 0.27Ω resistor at 37mA peak load.

$$V_{0.27\Omega} = 0.037 \times 0.27[mV]$$

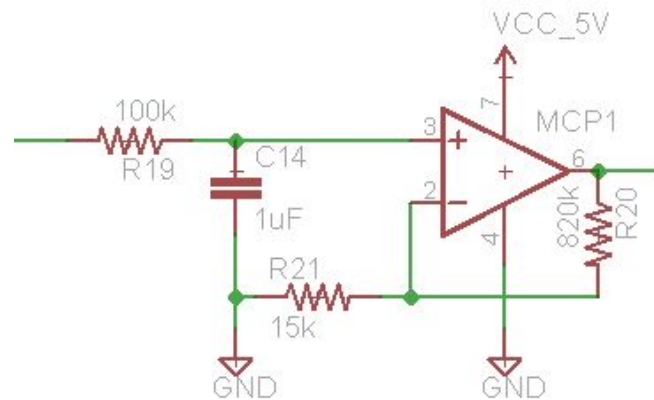
$$V_{0.27\Omega} = 10[mV]$$

The motor driver features a voltage output that is proportional to the current through the 0.27Ω resistor by a factor of 5.

$$V_{propi} = 5 \times V_{0.27\Omega}$$

$$V_{propi} = 50[mV]$$

Because sampling at that voltage would be too insensitive to motor strain, it is therefore required to amplify the motor current to a value halfway in-between Vcc and GND.



Op amp circuit to amplify Vpropi

$$V_O = V_{propi} \times \left(1 + \frac{R_2}{R_1}\right)$$

Therefore, an op amp in this non-inverting configuration was used to increase the accuracy.

Using an 820kΩ resistor for R2 and a 15kΩ for R1 we have a gain of 56. This ensures an output value of 2.8V for the ADC. The resistors were chosen in the tens of thousands in order to limit the current the can pass out of Vo of the op amp, since the max current rating is 2mA.

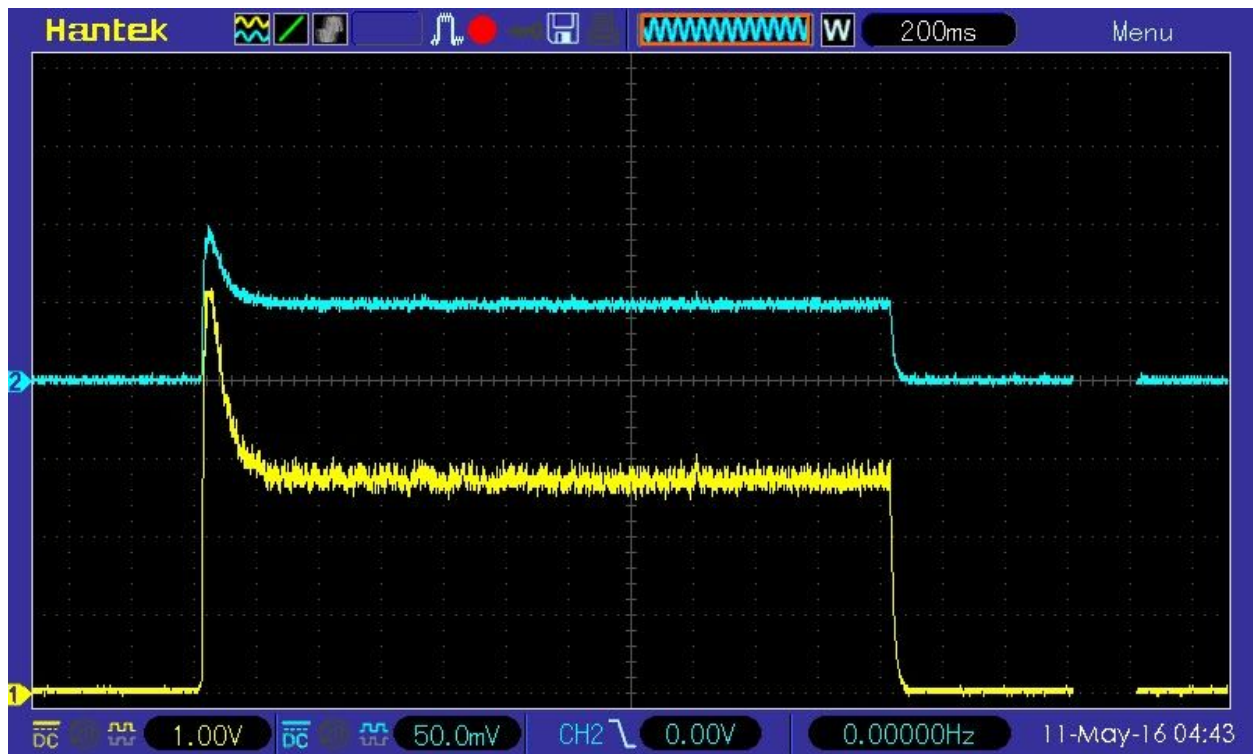
$$0.05 \times 56 = 2.8[V]$$

$$\frac{2.805}{56} = 0.050089$$

$$\left(\frac{0.000089}{0.005}\right) \times 100 = 1.79$$

$$\frac{100}{1.79} = 56$$

This also means that it is 56 times more accurate.



Gate opening: V_{out} for ADC (yellow - about 2.8V)

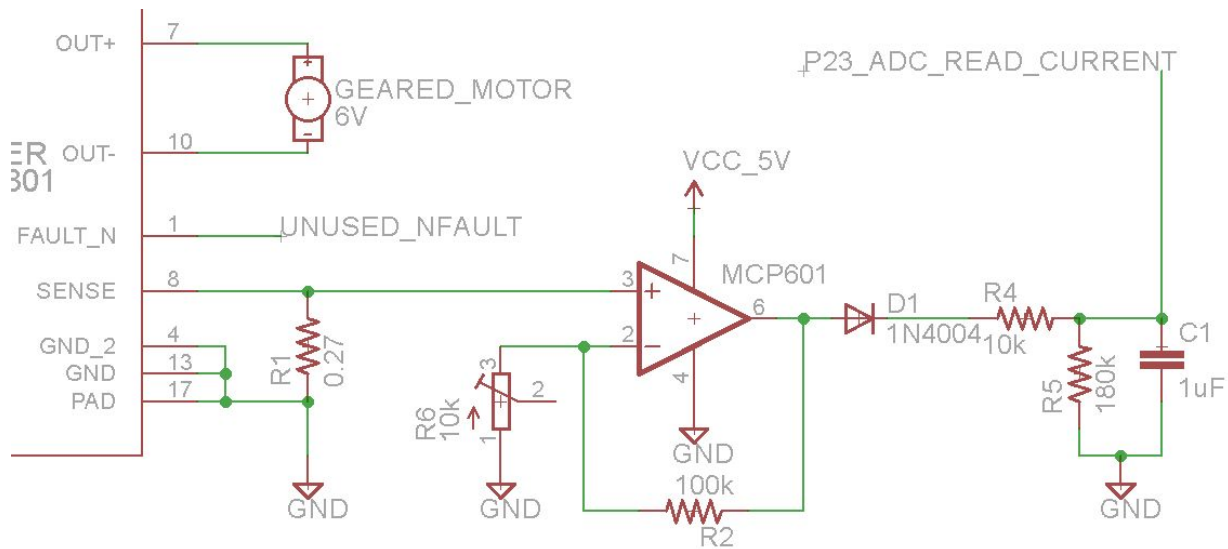
For the microcontroller to realise the true value of current (in this case 37mA), some conversions are needed. The value comes as a hex value between 0 and 1024. If 37 mA, the ADC would read 2.8V as a hex value of 573 or 0x23D.

To get that hex value to 37mA, one would need to divide by 15.5. For code efficiency, it is easier to divide by a power of 2. 16 is closest to 15.5.

Therefore, one can bitshift 4 bits to the right, since $2^4 = 16$, and bit shifting is equivalent in binary to dividing by 16 in the decimal system.

Unfortunately, during development, the internal difference-amplifier for V_{prop} or one of the boards broke. This was most likely due to overrated current through that pin.

An original plan was created to overcome this difficulty. A different circuit to the one proposed was developed, which included a diode, and a parallel RC circuit.



It was quite suitable, and very sensitive, but since the gain was, in essence, “zoomed in”, it had a limited current range to measure in. Anything above 70mA clipped the op amp to nearly 5V.

Therefore, for the purposes of this investigation, and for added simplicity, the originally proposed plan is used and discussed.



nSleep toggle switch

A toggle switch was used to switch between nSleep from the microcontroller and GND, in an effort to provide an emergency stop in case the microcontroller died while the gate is moving.

Any other capacitors in the motor controller schematic were straight from the datasheet.

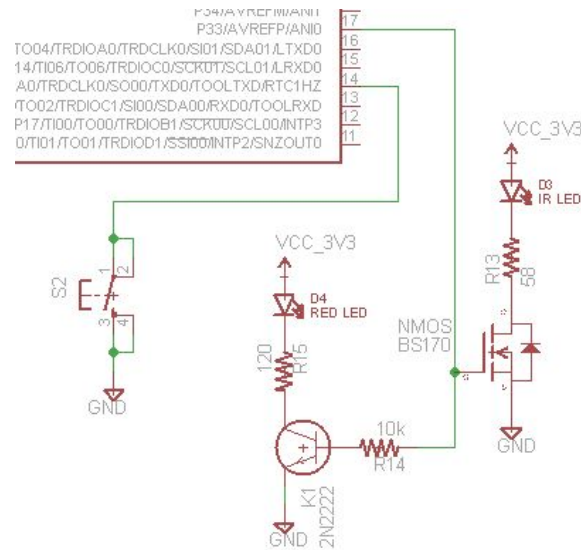
3.6 Infrared

An infrared interface was required to control the gate from a distance, encrypted with a unique address for each operator and an RC5 Manchester encoding protocol.

To design a transmitter and receiver, difficulty was encountered as the infrared remote and the controller board have independent processors, as well as no hardware debugger being available for the former.

To address this issue, an infrared transmitter/receiver LED diode pair, including a button, was temporarily constructed on the main controller board for testing purposes and to assist in debugging, thus creating a homogenous co-processing environment.

Transmitter



Infra-red transmitter

The equation of current for a saturated BS170 mosfet follows below. V_{TN} was taken as an approximate value from the datasheet.

$$i_{DQ} = K_n(V_{GS} - V_{TN})^2$$

$$3.3 = K_n(3.3 - 1.5)^2 R_D + 2 + V_{DS}$$

Unfortunately, the datasheet did not include the specifications for the width/length of the FET gate etc, or a known means to calculate the value K_n .

However, it was specified that the current passing through the IR LED should be approximately 20mA, and that V_{DS} should be about 0.2V (from the datasheet).

$$3.3 = (20 \times 10^{-3})R_D + 2 + 0.2$$

$$R_D = 55[\Omega]$$

$$R_D \approx 58[\Omega]$$

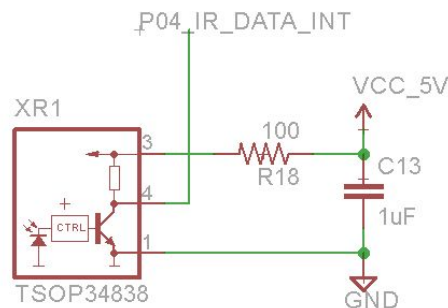
Another circuit was implemented for a red LED using a 2N2222 transistor. It is used for feedback to the user. If it stops flashing (when transmitting), then it would signify that the battery is flat.

The buttons are connected in pull-up resistor fashion to interrupts.

The remote uses a manchester encoding with a 38kHz carrier.

The remote is powered by a 3.7V, 1100mAH lithium battery.

Receiver



Infra-red receiver

$$R = 100[\Omega]$$

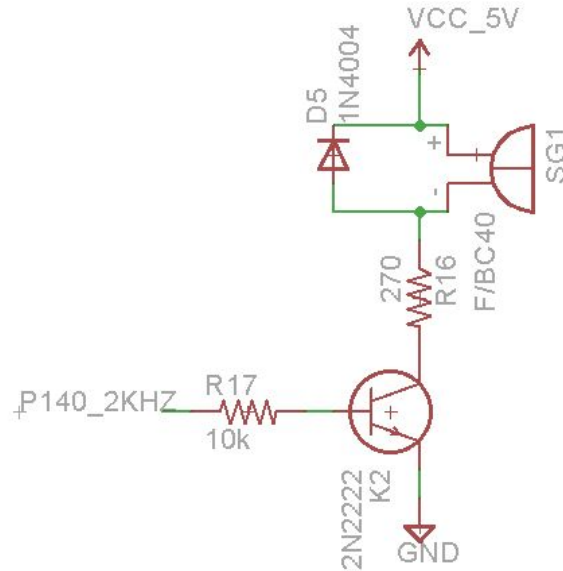
$$C = 1[\mu F]$$

The 100Ω resistor is to stabilise the input voltage from noise from the power supply. The 1μF capacitor is to smooth out noise.

3.7 Other peripherals



The PTC resettable fuse holds at 200mA and triggers at 350mA. It can stop up to 40A, with good thermal dissipation, since the on resistance is quite minimal.



Buzzer Circuit

For the buzzer,

$$V_{buzzer} = V_{CC} - V_{CE(sat)} - i_{CQ}R_C$$

$$2 = 5 - 0.2 - 20 \cdot 10^{-3}(R_c)$$

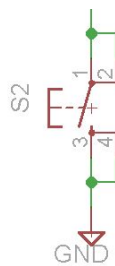
$$R_c = 270ohm$$

$$V_{enable-buzzer} = \frac{i_c}{\beta}R_B + V_{BE(on)}$$

$$5 = \frac{20 \cdot 10^{-3}}{100}(R_b) + 0.7$$

$$R_b = 21.5kohms$$

Rb was rounded down to 10kΩ to make sure that the transistor is in saturation.



Momentary button

All switches and buttons are connected using internal 22kΩ pull-up resistors in the microcontrollers.

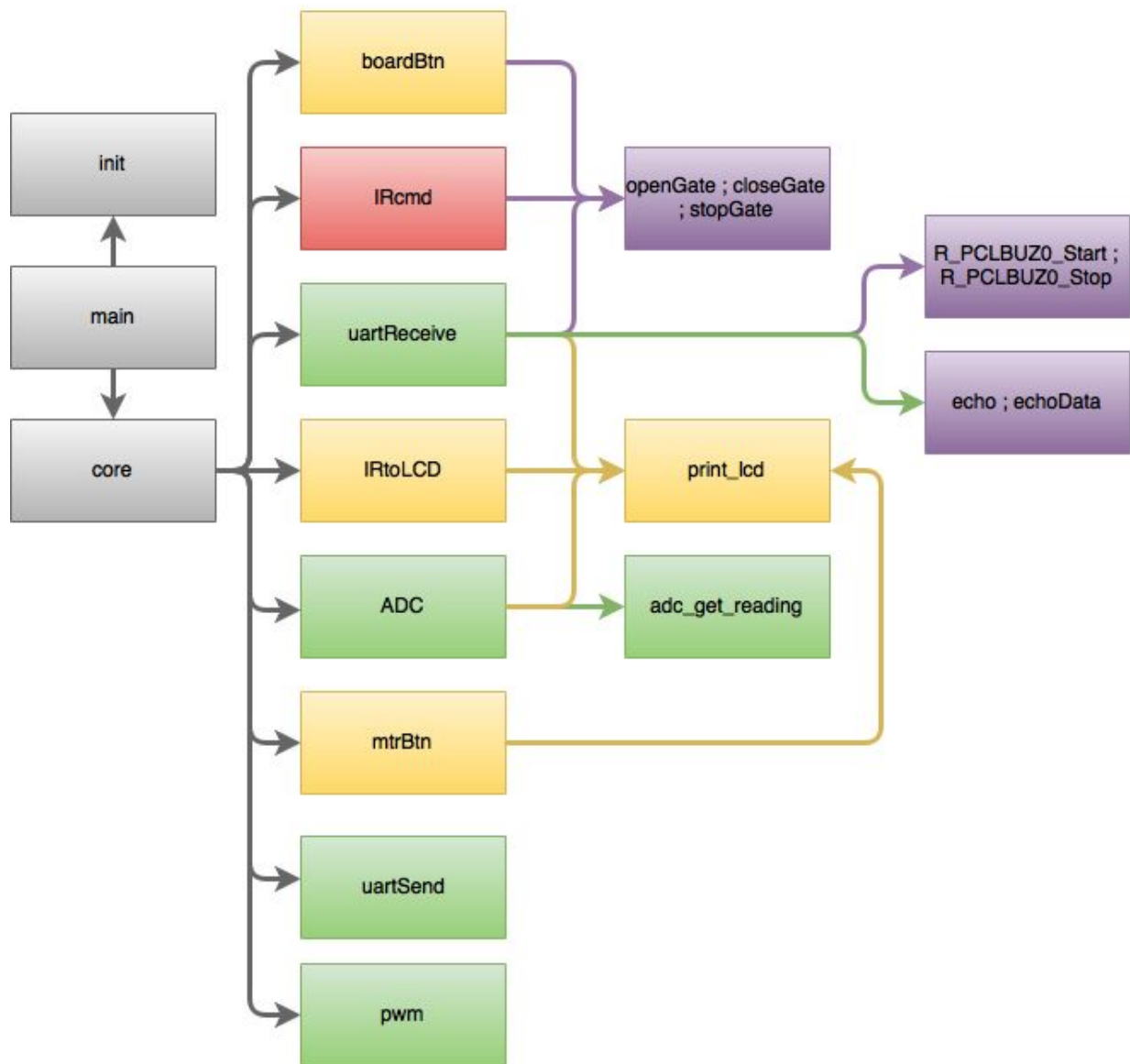
There exists an onboard 10k potentiometer on the G14 promotional board. It is used to control the motor speed. The motor speed can reach it's rated max speed at 6V DV by change the duty

cycle of the PWM from 0 to 44%. The max possible voltage is 15V on the gate controller, and 44% of 15V is 6V. This helped especially for the current sensing development.

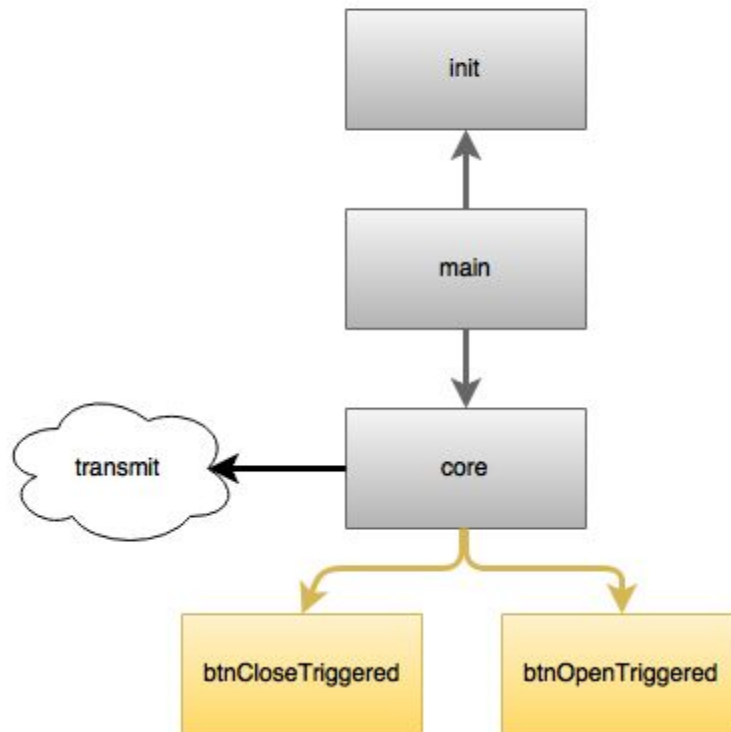
The toggle switch between the 5V regulator and the G14 was especially helpful in debugging the gate controller during development.

4. Programs

GateController.c functions flow diagram



RemoteController.c functions diagram



The whole program is written in a waterfall style, where it deals with interrupts in the main loop. In general, there were hardly any calculations, since the coding was written in an exact, logical manner. Many timers were used. NOPs were avoided when influencing a delay.

Gate Controller:

global.c

Function	Description
void init(void)	Function to initialise the program variables, timers, connections, interrupts etc.
void core(void)	Primary loop function, handles event flags (buttons pressed, commands sent/received)
void mtrBtn(void)	Function to process proximity switch presses. Stops gate if contact is detected and changes gate status
void boardBtn(void)	Function to process PCB button presses: open/close gate, estop, menu step (menu step not implemented)
void IRtoLCD(void)	Function to display last IR command to LCD
void uartReceive(void)	Function to process UART Receive flag. Executes commands corresponding to the received hex data.
void uartSend(void)	Function to simply reset the UART Send flag. Actual data transfer is handled by generated function [MD_STATUS R_UART1_Send(uint8_t * const tx_buf, uint16_t tx_num)]
void pwm(int cycles_per_second , int divisor)	Function to output Pulse Width Modulation signal to motor - motor's utilisation of the PWM is toggled by its nSLEEP. PWM = divisor/cycles_per_second %
void msDelay(int t)	Delay for t milliseconds with no interrupts
void echo(uint8_t hex)	Function to echo one Byte of data (hex) via UART to PC
void echoData(uint8_t hex , uint8_t hex2)	Function to echo two Bytes of data (hex1 , hex2) via UART to PC

void bufferToLCD(void)	Function to write data in the buffer to the LCD
void openGate(void)	Opens the gate
void closeGate(void)	Closes the gate
void stopGate(void)	Stops the gate (emergency stop)
void setRTC(void)	Sets the Real Time Clock to the last 5 Bytes stored in the UART Receive buffer [myBuffer]
void sendRTC(void)	Sends the current Real Time Clock value via UART
void buzzerBeep(uint8_t beeps)	Causes the buzzer to emit a series of n beeps at 20 millisecond intervals
uint8_t toHex(uint8_t decimal)	Converts Binary Coded Decimal to Hexadecimal
uint8_t toBCD(uint8_t hex)	Converts Hexadecimal to Binary Coded Decimal

adc.c

void ADC(void)	Function to handle the motor current reading and display.
uint16_t adc_get_reading()	Get instantaneous current value of motor as 16 bit value. Read via Analog to Digital Conversion.

infrared.c

void IRcmd(void)	Function to handle received Infra-red commands (use function below) Checks the address contained in data with [TNEladdr] in infrared.h before executing command contained in data
void receive(void)	Function to process received Infra-red data into possible commands

lcd.c

void startTMR0(int delay)	Starts the timer that forms the delay function of n delay microseconds
void delayNoInt(uint16_t delay)	Function to delay n delay microseconds
void writeByteLcd(uint8_t reg , uint8_t value)	This function writes 2 nibbles to the LCD. reg (BOOL) - 0 = register, 1 = data value (BYTE) - first the upper 4 bits and the 4 lower bits
void writeNibbleLcd(uint8_t reg , uint8_t nibble)	This function writes a nibble to the LCD. reg (BOOL) - 0 = register, 1 = data nibble (BYTE) - only the lower 4 bits are used
void initLcd(void)	This function initializes the LCD
void testLCDConnections(void)	Provided function - unknown use
void lcd_clear()	Clears the LCD
void print_lcd(uint8_t * message)	Prints a message to the LCD
void delay(uint16_t delay)	Approximate for-loop delay
void word_to_ascii(uint16_t word , uint8_t * lcd_word)	Converts Infra-red Manchester Encoding [word] to normal Binary format for LCD display [lcd_word]
void welcome(void)	Welcome the user on LCD. User name and student number stored in lcd.h [USERNAME]

Remote Controller

global.c

void init(void)	Function to initialise the program variables, timers, connections, interrupts etc.
void core(void)	Primary loop function, handles event flags (buttons pressed)
void transmit(void)	Transmits stored array of 14 bits via IR LED
void btnOpenTriggered(void)	Function to handle Open Gate button event - If pressed alone, open gate - If let go while Close Gate is not pressed, stop transmitting - If pressed while Close Gate is pressed, emergency stop - If let go while Close Gate is pressed, close gate
void btnCloseTriggered(void)	Function to handle Close Gate button event: - If pressed alone, close gate - If let go while Open Gate is not pressed, stop transmitting - If pressed while Open Gate is pressed, emergency stop - If let go while Open Gate is pressed, open gate

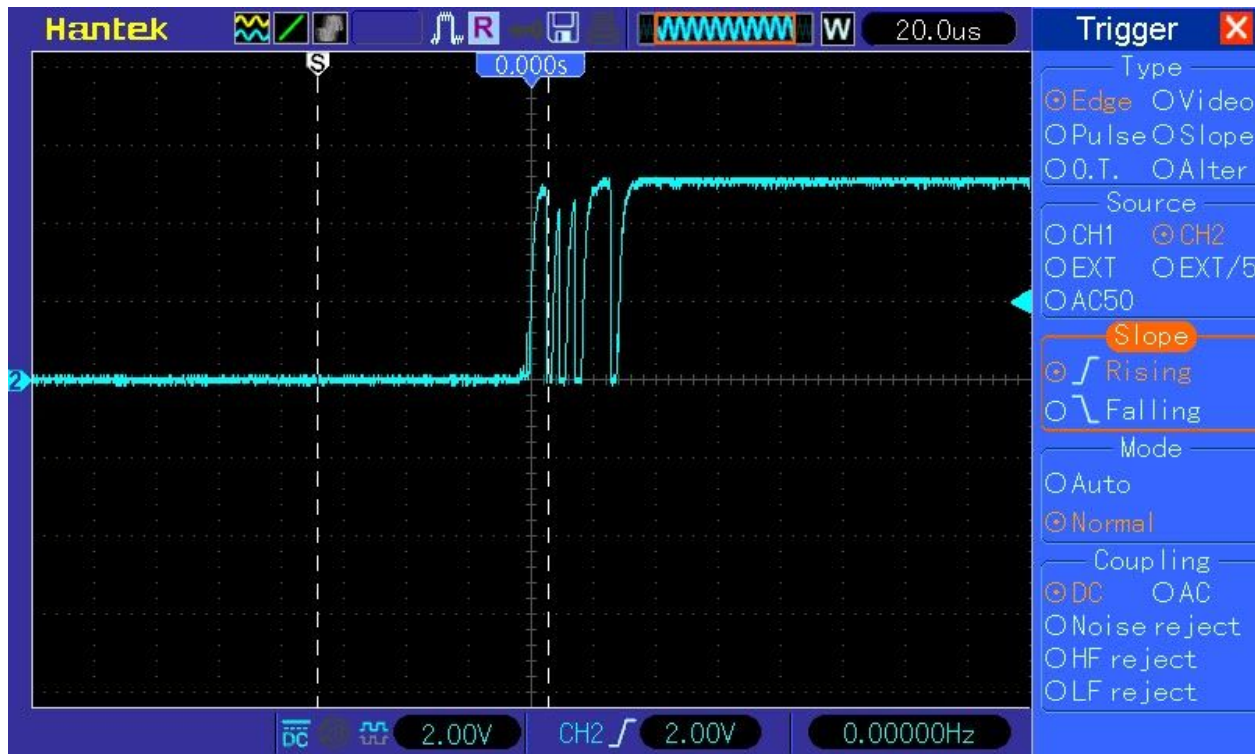
- src
 - event
 - event.c
 - event.h
 - flash
 - flash.c
 - flash.h
 - pfdl_asm.h
 - pfdl_types.h
 - pfdl.h
 - pfdl.a
 - system
 - adc.c
 - adc.h
 - buzzer.c
 - buzzer.h
 - core.c
 - core.h
 - debug.h
 - delay.c
 - delay.h
 - gate_controller.c
 - gate_controller.h
 - globals.h
 - infrared.c
 - infrared.h
 - io.c
 - io.h
 - lcd.c
 - lcd.h
 - math.c
 - math.h
 - motor_driver.c
 - motor_driver.h
 - pins.h
 - serial.c
 - serial.h
 - time.h
 - utility.c
 - utility.h
 - iodefine_ext.h
 - iodefine.h
 - r_cg_adc_user.c
 - r_cg_adc.c
 - r_cg_adc.h
 - r_cg_cg_user.c
 - r_cg_cg.c
 - r_cg_cg.h
 - r_cg_intc_user.c
 - r_cg_intc.c
 - r_cg_intc.h
 - r_cg_interrupt_handlers.h
 - r_cg_macrodriver.h
 - r_cg_pclbuz_user.c
 - r_cg_pclbuz.c
 - r_cg_pclbuz.h
 - r_cg_port_user.c
 - r_cg_port.c
 - r_cg_port.h
 - r_cg_rtc_user.c
 - r_cg_rtc.c
 - r_cg_rtc.h
 - r_cg_serial_user.c
 - r_cg_serial.c
 - r_cg_serial.h
 - r_cg_timer_user.c
 - r_cg_timer.c
 - r_cg_timer.h
 - r_cg_userdefine.h
 - r_cg_vector_table.c
 - r_hardware_setup.c
 - r_main.c
 - r_reset_program.asm

File structure (Daniel). Folders were used.

5. Measurements and results

It was observed that the natural frequency of the motor is actually 67.5Hz.

The 5V rails are actually 5.075 volts.

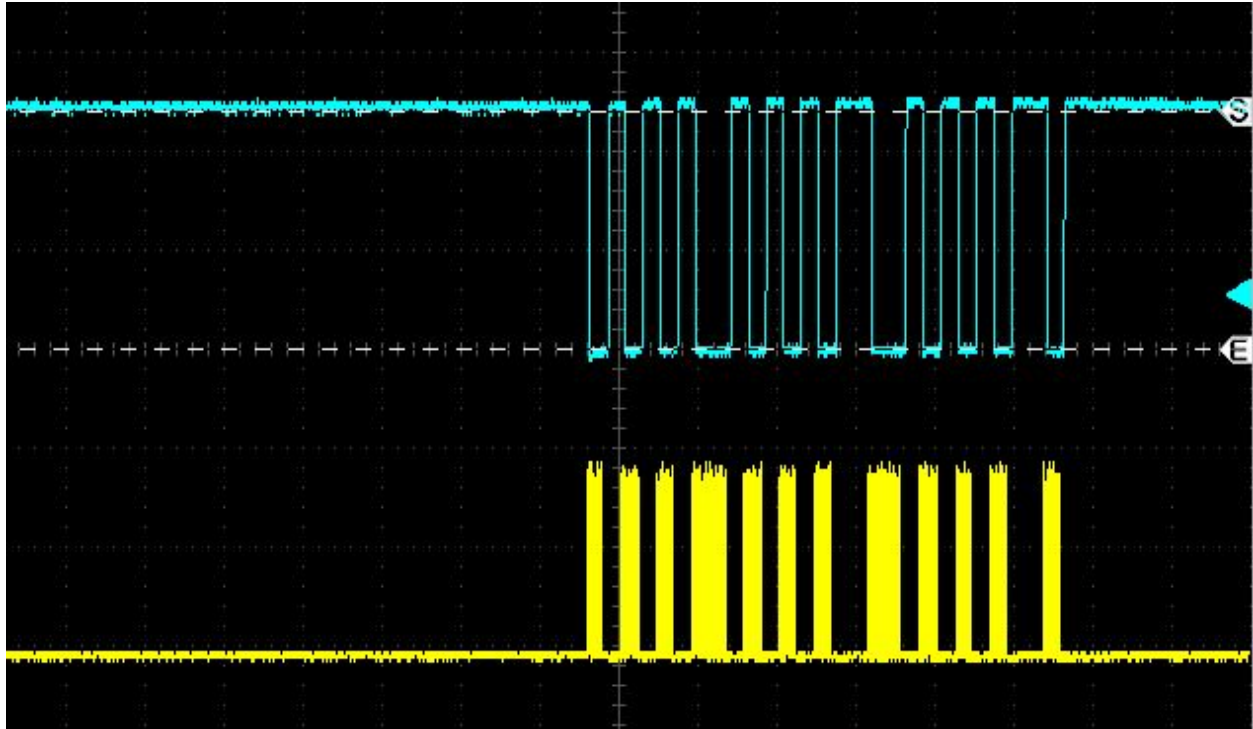


Bouncing on buttons

Hardware debouncing was performed on the buttons, with an RC LPF with a time constant of 0.1 seconds. $R = 100\text{k}\Omega$ and $C = 1\mu\text{F}$. Unfortunately, as we can see here, there is still bouncing on the mechanical switches for about $25\mu\text{s}$. It was determined that hardware debouncing does not work. Therefore, because hardware interrupts work at a very fast rate, it is not suitable. On the other hand, as is implemented in the code, polling the button at a rate of 1kHz has a minimal effect on the system, and bouncing is picked up less than 5% of the time.

The motor runs at about 37mA.

For the buzzer, $V_{\text{CE}} = 0.2\text{V}$ as expected (saturation). The current is $4.8/120 = 40\text{mA}$, which is not as expected. This is because, when the buzzer is in action, its actual voltage is close to zero due to the low impedance/ESR, and not 2V as stipulated in the datasheet.



IR transmitter gate signal (yellow) and IR received signal (blue)

An RC5 Manchester encoded bit varies from 870 - 890 μ s. The PWM rate out of the transmitter is exactly 38kHz. Interestingly, there is a slight phase shift between the transmitter and receiver. This can be attributed to a propagation delay in the amplification process in the IR receiver.

The current through the IR LED is $(2.32 - 0.04)/58 = 39.3\text{mA}$. This is double what was expected, since the on voltage of the IR LED is indeed much smaller than 2V as expected. The on voltage of the IR LED is measured to be $2.88 - 2.32 = 0.56\text{V}$. Also, V_{ds} is even lower than 0.2 volts as previously thought. It is 40mV. This means that R_d should have been at least 100 ohms.

	Daniel	Tristan	Comment
IR LED current	36mA	39mA	
Motor current	25mA	37mA	Different motor speed
Motor current strain	35mA	50mA	
Buzzer current	40mA	38mA	
Renesas G14	40mA	40mA	
Total current (idle)	105mA	55mA	Extras LEDs

Table depicting differences in measured values between two boards

6. Conclusion

The development time for this project was longer than expected. An Arduino would have been a more ideal processor choice, compared to a Renesas board, since there is such a strong community base out there for it - although cost might be a factor. Renesas has great documentation, but many students struggled with strange errors and values and forums cannot be consulted for most such errors, which increases development time.

Also, due to the budget being a bit too low, as it is a prototype, many students' fuses, ICs, controllers and components broke in the development process, and without having a large surplus to mitigate these problems, it lengthened the development time greatly when a part had to be ordered. Delivery time is about a week.

Regarding the gate controller system, it was found that hardware debouncing does not work as expected.

Resettable fuses are recommended, since they have a very quick response time in surge conditions.

A 16x2 LCD display is recommended as more information can be displayed. It is also recommended to have a backlight especially if testing in a dark place, or at night.

The resistors for the buzzer and the IR transmitter may need to be altered slightly so as to lower the current. This would contribute to power saving.

Due to the efficient design approach of the project, all the implemented functions of the gate controller and remote, performed optimally as expected.

Appendix

A. User Manual

How to operate the Gate Controller and Remote Controller

Connect the Gate Assembly to the Controller Board

The Gate Motor should be connected with appropriate polarities to the Controller Board's Gate Output port (see schematic)

Power the Gate Controller Board and Remote Controller

The Gate Controller Board has a 12V DC input voltage port (see schematic), and this should be used to power the board.

The micro-USB port on the processor is for hardware debug purposes only.

DO NOT connect the micro-USB and 12V DC input simultaneously, as this could result in a blown fuse (see schematic) or damaged processor.

The Remote Controller requires a 3.3V battery to operate.

Normal Mode vs Test Mode

The Gate Controller processor runs in one of two modes, which can be switched between using PC debugger software, for which the board must be connected to the PC via a UART cable (see UART port on schematic).

The processor starts in Normal Mode after displaying the user welcome message.

The following functions can be performed in each respective mode:

Normal Mode

Remote Control

- Open the gate using upper button (will sound a buzzer tone)
- Close the gate using lower button (will sound a buzzer tone)
- When the gate is opening or closing, the incident as well as the motor current reading will be displayed on the board LCD
- When the gate is fully opened or closed, LCD will read "Opened" or "Closed" respectively
- Hold in both buttons to emergency stop gate

Board

- Emergency Stop (lower button)
- Use menu up/down buttons to scroll through events

Test Mode

(Must be switched to via PC debug program command while connected via UART)

Board

- Last received infrared command will be displayed on LCD
- Open the gate with the left button
- Close the gate with the right button

PC Debug Software and UART connection

Using the buttons in the program:

- Read or set the processor Real Time Clock (display in software)
- Enable/Disable buzzer tone (constant tone)
- Enter text into the *Ascii Input* field and select *Send Ascii* to transmit the data to the board
- Select *Display on LCD* to have the previously sent data displayed on board LCD screen
- Open/Close gate
- Read motor current (display in software)
- Read gate status:
 - Unknown
 - Opened
 - Closed
 - Emergency stop
- Read last received infrared command (display in software)
- Read latest event
- Erase datalog

B. Technical Specification

Parameter	Nominal value	Minimum rating	Maximum rating	Unit
Main board voltage	12	9	15	V
Main board current	50	40	100	mA
IR transmitter	3.3	3	5	V

Pinout

G14 Gate controller

CN2

Pin	Signal Name	RL78/G14 Pin Name	Function
1	GN D	VSS / EVSS	Ground
2	VD D	VDD / EVDD	5V external input
8	P15	P15_SCK20_Z_SCL20_TRDIO D0	This is the 40kHz pwm enable for the motor (directed to the enable pin of the DRV8801)
9	P16	P16_TIO1_TO01_INTP5_TRDI OC0_IVREF0	This interrupt pin is triggered by data from the IR receiver

10	P17	P17_TIO1_TO02_TRDIOA0_TRDCLK0_IVCMP0	This is connected to a button which closes gate upon pressing
11	P55	P55_TIO1_TO01_INTP5_TRDI OC0_IVREF0	This is connected to a button which shows the previous event upon pressing
12	P54	P54_TIO1_TO01_INTP5_TRDI OC0_IVREF0	This is connected to a button which shows the next event upon pressing
13	P53	P53_TIO1_TO01_INTP5_TRDI OC0_IVREF0	This is connected to a button which will stop the gate from moving upon pressing (as if an emergency stop)
14	P52	P52_TIO1_TO01_INTP5_TRDI OC0_IVREF0	This is connected to a button which opens gate upon pressing
15	P51	P51_INTP2_SO00_TXD0_TOO LTXD_TRGIOB	<i>For testing.</i> A button triggers this hardware interrupt which in turn sends a test manchester string.
23	P76	P76_KR6_INTP10_RXD2	LCD Register select
24	P75	P75_KR5_INTP9_SCK01_SCL01	LCD Read /Write
25	P74	P74_KR4_INTP8_SI01_SDA01	LCD Enable
26	P73	P73_KR3_SO01	LCD Data channel 3
27	P72	P72_KR2_SO21	LCD Data channel 2
28	P71	P71_KR1_SI21_SDA21	LCD Data channel 1

29	P70	P70_KR0_SCK21_SCL21	LCD Data channel 0
----	-----	---------------------	--------------------

CN3

Pin	Signal Name	RL78/G14 Pin Name	Function
1	GND	VSS / EVSS	Ground
2	VDD	VDD / EVDD	5V external input
3	P140	P140_PCLBUZ0_INTP6	2kHz buzzer output
7	P02	P02_ANI17_SO10_TxD1	Serial output to external test program
8	P03	P03_ANI16_SI10_RxD1_SDA10	Serial input from external test program
10	P130	P130	<i>For testing.</i> Here is where a manchester string is output to a built-in IR led.
11	P20	P20_ANI0_AVREFP	<i>Unused</i> ADC input
12	P21	P21_ANI0_AVREFM	<i>Unused</i> ADC input TODO: use potentiometer to cycle through menu.
13	P22	P22_ANI2_ANO0	This reads a potentiometer built-in the RL78 G14. It controls the speed of the gate motor.
14	P23	P23_ANI3_ANO1	This reads in a current value from 0 - 100mA, converted to a value

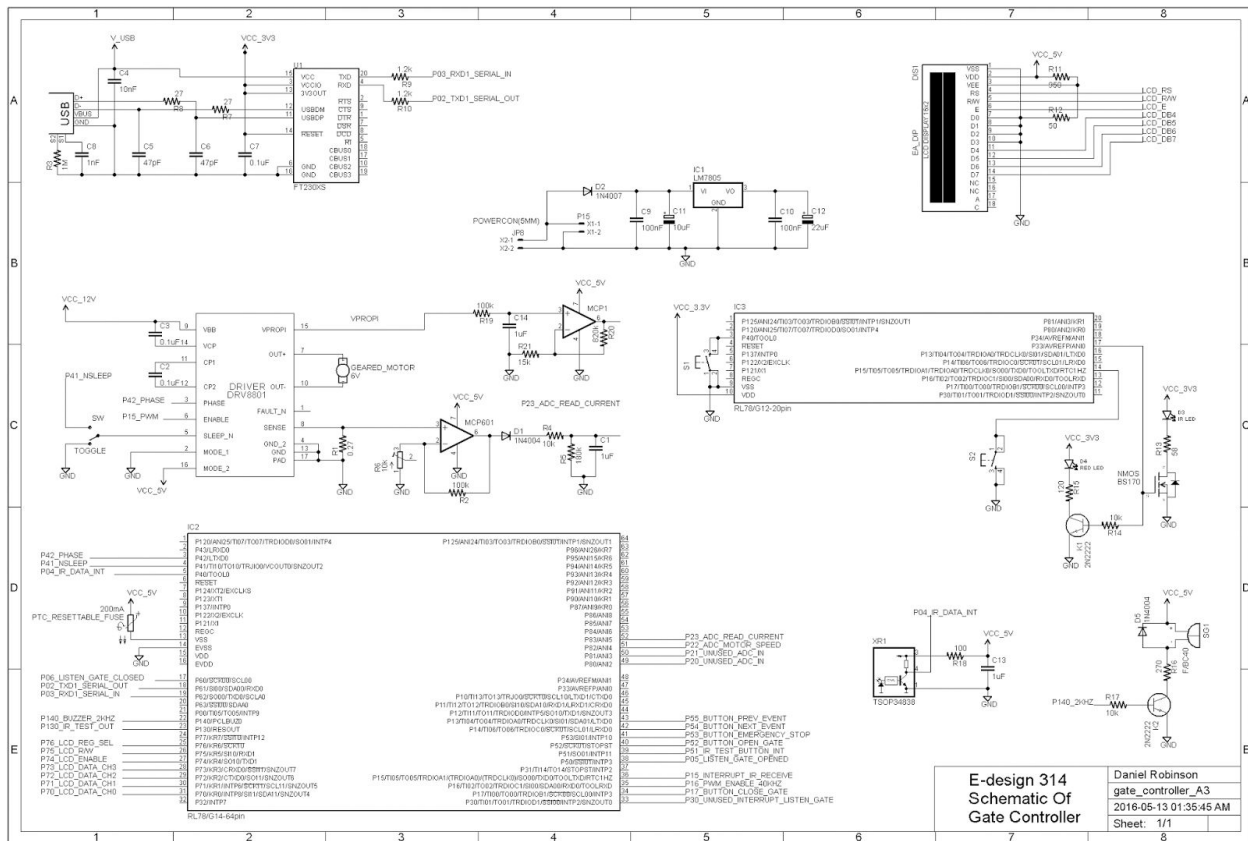
			between 0 - 5 V, with 10 bit accuracy.
22	P42	P42	This digital output controls the phase/direction that the gate motor is turning.
23	P41	P41	This digital output can sleep the motor driver (nSleep).
27	P06	P06	Digital in which listens for when the gate closes/is closed (port is polled ~ debouncing).
28	P05	P05	Digital in which listens for when the gate opens/is opened (port is polled ~ debouncing).
29	P30	P30/INTP3/RTC1HZ/SCK00/SCL00/TRJ00	<i>Unused.</i> Hardware interrupt which listens for when the gate opens/closes.

G12 Infra-red transmitter

Pin	Signal Name	RL78/G14 Pin Name	Function
3	P41	P41_ANI22_SO01_SDA01_TI02_TO02_INT01	Open gate button
9	GND	VSS	Ground
10	VDD	VDD	3.3V external input

14	P13	P13_ANI19_TI00_TO00_INTP2	Close gate button
17	P10	P10_ANI16_PCLBUZ0_SCK00_SCL00	IR Led output

C. Circuit Diagrams



Small-scale version of entire A3 schematic

D. Electronic submission:

This report can be found on

<https://goo.gl/RYua6i>

Code and API can be found on

<https://github.com/daniel-leonard-robinson/Wireless-Gate-Controller>

<https://github.com/gorrox/E-Design-314-Motorised-Gate>

Symbols and Abbreviations

- ADC - Analogue to Digital Converter
- PWM - Pulse Width Modulation
- PSU - Power Supply Unit
- PC - Personal Computer
- GND - Ground
- LPF - Low Pass Filter
- IC - Integrated Controller
- NFC - Near Field Communication
- NOP - No Operation
- PTC - Positive Temperature Coefficient
- ESR - Equivalent Series Resistance

Bibliography

1. Texas Instruments. 2015. DRV8801 DMOS Full-bridge Motor Driver.
2. Vishay. 2003. IR Receiver Module TSOP348. Rev 5. Vishay Semiconductors
3. Fairchild. 2011. 2N3904 Data sheet. Fairchild Semiconductor Corporation
4. Vishay. 2001. N-Channel MOSFET T BS170. Vishay Semiconductors
5. Bright LED Electronics. BIRNM23C2DS. Bright LED Electronics Corporation
6. Renesas. 2013. RL78/G12 User's Manual: Hardware. Rev 2.00. Renesas Electronics
7. Renesas. 2012. RL78/G14 Demo User Manual: Hardware. Rev 1.00. Renesas Electronics
8. Smit, W and Treurnicht, J. 2016. Design (E) 314 Lecture x. Stellenbosch: Stellenbosch University
9. Treurnicht, J. 2016. Design (E) 314 Lecture x. Stellenbosch: Stellenbosch University
10. Treurnicht, J. 2016. Practical Week x. Stellenbosch: Stellenbosch University