***DyckCo* Wireless Remotely Controlled Home System Technical Manual**

Prepared by: Trevor Cook
Dustin Dyck

Date: April 16, 2007

DyckCo Wireless Remotely Controlled Home

Technical Manual
Table of Contents

INTRODUCTION

This technical manual describes a Microcontroller system solution for a low power wireless remotely controlled home system. This design implements RF600E Keeloq based encoders for the communication encryption, microcontroller PIC16C877A for signal monitoring and input and output controls, RF600D Keeloq based decoders for the communication decryption. Other support logic is included, such as a battery back-up circuit for the door lock, a simple single stage lead acid battery charger, and two electromagnets, but the main focus of the application is the implementation of Microchip Keeloq based security and PIC microcontroller with R.F. Solutions RF devices for a complete solution.

APPLICATIONS

Low power RF wireless systems are becoming more common as the technology makes the devices smaller and more powerful. With an increase of wireless communication, more messages are bouncing through the air, increasing the need for security and control over who gets what message, and what device should respond to which message. Low power devices usually transmit less than 1mW of power and do not require user licenses for operation. These systems operate over distances of 5 to 100 meters, depending on the application.

In many of these systems, different levels of security are required. The level of security required is dependent on the application and customer demands. The most vulnerable part of these systems is the actual RF signal itself. An RF signal could be captured out of the air and re-transmit by someone near to the device responding, and this could allow an unauthorized person complete control. Using security with code-hopping encryption is a secure solution so even if a code is captured, the same code is not used twice, and therefore the re-transmitted code is no longer valid to operate the system.
The Keeloq code hopping system makes code scanning impossible. The 32-bit encrypted

portion of the message provides for more than 4 billion code combinations.

The Keeloq encoders and decoders require only push buttons, battery/power, and RF transmitter/receivers to operate. They are also created to interface with PIC microcontrollers, allowing the decoders and encoders to be integrated to create system functionality.

SYSTEM OVERVIEW

Some specific design goals for this low power system were:

- A complete wireless solution
- Secure RF transmissions
- Line power operation of devices in the home so there is no need of operational batteries
- Microcontroller based system
- System remote controlled bi-directional communication

The three main hardware components which comprise this wireless remote controlled home system are:

- A Wireless Remote.
- 2 PIC controlled relay modules.
- 1 PIC controlled H-Bridge module

The PIC controlled relay modules in this system are:

- A voltage controller to turn a light on and off.
- An electromagnetic lock controller to lock or unlock a door.

The PIC controlled H-Bridge module is:

- A motor controller to open and close a door.

OPERATION OF THE SYSTEM

This system is designed to be controlled by a single remote control. The remote control has 3 buttons and 4 LEDs. Each button is labeled with a picture of the command it controls. The button with a picture of a light bulb turns a light on or off. The button with a picture of a padlock locks or unlocks the door. The button with a picture of an open door opens or closes the door.

Figure M-1 – DyckCo Remote Control Production Mock-Up

The first three LEDs are the notification lights responding to acknowledgement signals from whatever devices acknowledge executing a command from the remote. The fourth LED is lit whenever the remote is transmitting a command.

Example Usage

1. The light is currently off in the house.
2. Button 1 is pressed on the ***DyckCo*** Remote Control.
3. The fourth LED flashes to show remote is transmitting.
4. An LED on the ***DyckCo*** Voltage Controller turns on to show it has received the command.
5. The light turns on in the house.
6. The transmitting LED lights up for 5 seconds showing the ***DyckCo*** Voltage Controller is sending an acknowledgement signal to the ***DyckCo*** Remote Control.
7. On the ***DyckCo*** Remote Control, the first LED lights up for 2 seconds to advise the user the ***DyckCo*** Voltage Controller responded.
8. After 5 seconds the user can press a button again on the ***DyckCo*** Remote Control.

Technical Example Usage

1. The user presses button 1 on the ***DyckCo*** Remote Control to turn the light on.
2. The circuit is closed, sending a logic high through the switch to the input pin on the PIC16F877A microcontroller.
3. PORTC pin C0 receives a logic high from the button 1 press.
4. The PIC's programming processes this input, and creates an output signal.
5. PORTD pin D0 is set logic low by the program, and this logic low is sent to input pin S0 on the RF600E encoder.
6. Assuming the Encoder's serial number is 1234567, an 8-byte data word is created consisting of the serial number, and a character representing which input pin (S0-S3) received a logic low. In this case, the message is "1234567A".
7. This data message is changed into a 67-bit Keeloq encrypted message, and is transmitted via CMOS/TTL through the output pin 6 to the transmitter.
8. The AM-RT4-433 transmitter receives the signal on its input pin 3 and transmits it through the antenna on output pin 4.
9. The antenna on the ***DyckCo*** Voltage Controller receives the signal, and sends it to the input pin 3 on the AM-HRR30-433 mini receiver. The receiver then outputs this signal on pin 9 to the input pin on the decoder.
10. The RF600D decoder receives an input signal on pin 9 and decrypts the message.
11. The RF600D decrypted message is 1234567A. If the decoder is programmed to listen to encoder 1234567 it will process the button press, otherwise it will discard the message.
12. The ***DyckCo*** Voltage Controller RF600D decoder is programmed to listen to the RF600E encoder serial number 1234567. The message is A, therefore the decoder outputs a logic high on output data pin OP1.
13. The PIC16F877A microcontroller on the ***DyckCo*** Voltage Controller receives a logic high on pin C0.

14. The programming triggers the relay to the opposite state it is currently. Since the light is currently off, the G6BK-1114P relay has its reset coil triggered. Pin D4 produces a logic low and pin D5 produces a logic high to trigger the set coil on the relay to complete the circuit and allow the light to turn on.
15. PORTE pin E0 outputs a logic high for 1 second to acknowledge the command from the ***DyckCo*** Remote Control.
16. The PIC16F877A microcontroller then outputs a logic low on PORTD pin D0 to input pin S0 on the RF600E encoder. Assuming the encoder's serial number is 7654321, the encoder creates an 8-byte message of "7654321A" which is then encrypted and output through pin 6 to input pin 3 on the AM-RT4-433 transmitter.
17. The transmitter then outputs the signal on output pin 4 to the antenna.
18. The ***DyckCo*** Remote Control's antenna receives an input signal to pin 3 of the AM-HRR30-433 receiver. This signal is then output on pin 9 to the input pin 9 of the RF600D decoder.
19. The message is then decrypted by the decoder. This decoder is programmed to listen to encoder 7654321, and receives the message A. The decoder then sets data output pin OP1 logic high.
20. The PIC16F877A microcontroller receives a logic high on PORTC pin C4. The program then decides which notification LED should be sent a logic high.
21. PORTD pin D4 is set logic high, and notification LED 1 is turned on for 2 seconds to show the ***DyckCo*** Voltage Controller has acknowledged the command issued by the ***DyckCo*** Remote Control.

SYSTEM DESCRIPTION

The following sections provide a greater in depth look into each of the three main hardware components.

WIRELESS REMOTE

The wireless remote provides for:

- Transmitting of RF signals
- Monitoring of remotely initiated RF signals
- User interface via the 3 button keypad
- Visual feedback via the 4 LEDs

The wireless remote can be functionally divided into 3 main categories

1. R.F. Solutions Keeloq RF600E encoder interface and AMRT4 transmitter
2. R.F. Solutions Keeloq RF600D decoder interface and AMHRR30 receiver
3. PICmicro PIC16C877A interface and LED notification circuit.

Wireless Remote Operation

The most important task of the Wireless Remote is the creation and transmission of the output data of the RF600E encoder. The encoder is capable of creating a 67-bit encrypted signal based on the encoder's 7 byte serial number and 1 byte command. The encoder receives an input logic low from the microcontroller on one of its 4 input pins S0 to S3. This command is then encrypted and sent via the output pin 6 via CMOS/TTL, to the input pin 3 of the AMRT4 transmitter. This transmitter broadcasts the message at 433.92MHz through its output pin 4 via the antenna to a range of 70m.

After the signal is sent, the AMHRR30 receiver listens at 433.92MHz through the antenna connected to its input pin 3 to a range of 50m. If an acknowledgement signal comes back from one of the device modules, this message is sent via the receiver's output pin 9 into the RF600D decoder's input pin 9. The 67-bit message is decoded, and the decoder decides if the serial number of the encoding device is one that it should listen to. If the answer is yes, the decoder transmits the appropriate command through its 4 output pins OP1-OP4 to the PIC microcontroller. If the answer is no, the message is discarded.

The PIC16F877A microcontroller is the brains of this remote. It uses PORTC as an input port (pins C0-C7), and PORTD (pins D0-D7) as an output port. The user presses one of three buttons connected to input pins C0, C1, and C2, which produces logic high into the PIC. The PIC then executes the appropriate commands depending which pin was pressed, and sends a logic low output via pins D0, D1, or D2 to the RF600E encoder input pins S0, S1, or S2. The PIC then listens for input from the RF600D decoder's output pins OP1, OP2, or OP3 on input pins C4, C5, and C6. If a logic high is received on one of these pins, the PIC then sends a logic high to one of the 3 output LEDs connected to output pins D4, D5 or D6. These LEDs tell the user which device acknowledged receiving the button press command issued.

Microcontroller Controlled Relay Modules

The most important task of the Microcontroller controlled relay modules are the receiving and decryption of the data input from the RF600D decoder. The AMHRR30 or AMHRR3 receivers listen at 433.92MHz through the antenna connected to their input pin (3) to a range of 50m. A signal is sent from the receiver's output pin 9/14 to the input pin 9 of the RF600D decoder, decrypted, and the decoder ascertains whether or not it is supposed to accept commands from the encoder which sent the message. If the serial number of the encoding device is one that it should listen to, the decoder transmits the appropriate command through output pins OP1, OP2, or OP3 to the PIC microcontroller. If the answer is no, the message is discarded.

After the signal is received, and the command processed, an acknowledgment signal is transmitted back to the remote to advise the user the command was accepted and responded to. The encoder receives an input logic low from the microcontroller on input pins S0, S1, or S2. This command is then encrypted and sent via the output pin 6 via CMOS/TTL, to the input pin 3 of the AMRT4 transmitter. This transmitter broadcasts the message at 433.92MHz through its output pin 4 via the antenna to a range of 70m.

The PIC16F877A microcontroller is the brains of this remote. It uses PORTC as an input port (pins C0-C7), and PORTD (pins D0-D7) and PORTE (E0-E2) as output ports. The PIC then executes the appropriate commands depending which pin, C0, C1, or C2 was triggered, turns an LED via Port E pin E0, and triggers a relay coil, either Set (D5, D6) or Reset (D4, D7). The PIC then sends a logic low output via pins D0, D1, or D2 to the RF600E encoder's input pins. This message is transmit through the AMRT4 transmitter as an acknowledgement signal to the ***DyckCo*** Remote Control.

DyckCo Voltage Controller's Relay Operation

The Voltage Controller uses 1 relay to turn whatever is plugged into it on or off depending if the Set or Reset coils on the relay were triggered by a logic high from the PIC's output pins D4 or D5 respectively. The Relay is connected to +120 volts from the wall, and when the relay coil is Set, the circuit is completed, and the light, or whatever is plugged into this device, will be turned on. When the relay coil is reset, the circuit is opened, and the voltage is interrupted, so the light plugged into this device will turn off. Our usage in this project for this device is to control a light, so a 60W bulb uses 500mA, and a 100W bulb uses less than 1A.

DyckCo Door Lock Controller's Relay Operation

The Door Lock modules uses two relays, with pins D4 and D7 as Set coil outputs, and pins D5 and D6 as Reset coil outputs. Relay 1, PIC output pins D4 and D5, are connected to +5 volts separately. Relay 2, PIC output pins D7 and D6, are connected to +5 volts separately. Both relay's set coils are triggered when the user wants to unlock the door, and both relay's reset coils are triggered when the user wants to re-lock the door. The door lock mechanism uses two electromagnets powered with 5V at 2A.

DyckCo Door Motor Controller's H-Bridge Operation

The Door Motor module uses an H-Bridge, with pin D4 as a counter-clockwise (open door) output and pin D5 as a clockwise (close door) output. One side of the H-Bridge is triggered, and the door opens, the other side is triggered, and the door closes. The motor for the door opener operates with +5V/0V and 250mA.

Selection of Devices

Description of Devices

Transmitter

The R.F. Solutions AM Hybrid transmitter module provides a complete RF transmitter which can be used to transmit data at up to 4 kHz from any standard CMOS/TTL source. The modules are compatible with R.F. Solutions range of AM receivers to provide a complete solution. (DS013)

It is designed to operate with a supply voltage of 2-14V at an RF frequency of 433MHz. They have a transmit range up to 70m.

These transmitters are also designed to work with the RF600E Keeloq encoder, produced by R.F. Solutions, see Datasheet DS600 on the RFSolutions.co.uk website.

Receivers

The R.F. Solutions range of AM Super-regenerative receiver modules are compact hybrid RF receivers which can be used to capture undecoded data from any AM transmitter. They produce a CMOS/TTL output, and require connections to power and antenna only. (DS014)

There are two RF receivers in use in this system. AM-HRR30-433 is a miniature AM super-regenerative receiver, see Datasheet DS014. The AM-HRR3-433 is an AM super-regenerative receiver, see Datasheet DS016. Both of these devices were produced by R.F. Solutions. They use a supply voltage of 5V and operate at an RF frequency of 433MHz. They have a receiving range up to 50m. They are manufactured to be compatible with the AM Transmitters (AM-RT4/5-433) produced by R.F. Solutions.

These receivers are also designed to work with the RF600D Keeloq decoder, produced by R.F. Solutions, see Datasheet DS600 on the RFSolutions.co.uk website.

Encoder

The R.F. Solutions Keeloq RF Encoder uses the highly secure Microchip Keeloq Code Hopping Protocol to encrypt an 8 byte message into a 67 bit encrypted message. The RF600E uses a source voltage between 2 to 6.6V, with automatic battery level monitoring, and 'Manchester' modulation. (DS600)

It requires the addition of input switches and RF circuitry for use as the transmitter in the end application. Transmission is automatic without user intervention. The RF600E will wake up upon detecting a switch closure, and then delay approximately 6.5ms for switch debounce. The encrypted or hopping code portion of the transmission will change every time a button is pressed, even if the same button is pushed again. (DS600)

The data format automatically includes a pre-amble, synchronization header, followed by the encrypted and fixed code data then a CRC check. The actual packet size is 67 bits. Each transmission is followed by a guard period before another transmission can begin.

The data encryption provides up to four billion changing code combinations and includes the function bits (based on which buttons were activated) along with other data and synchronization information that the decoder uses. (DS600)

The transmitter serial number: is made up of 7 ASCII characters from the following group: 0 1 2 3 4 5 6 7 8 9 A B C D E F. The most significant serial number digit is transmitted first. This provides a total of 28 million possible serial numbers. The Keeloq button status is transmitted as an ASCII character in the range of A to O or if the low battery bit is asserted (encoder battery is low) then in the range a to o. Keeloq data bits S0, S1, S2 and S3 then make up the ASCII byte least significant bits D0, D1, D2, and D3.

If a single encoder button is pressed, say S0, then character A will be sent, or a if low battery.

Decoder

The RF600D has the capability to learn up to 7 unique RF600E transmitters. With the addition of an EEPROM memory device, this is increased to 48 RF600E encoders. (DS600)

The RF600D is connected directly to the data output of a radio receiver module and upon detection of a valid packet; it will decode the Keeloq encrypted data stream. When an address match occurs, the RF600D asserts its 4 digital outputs. These will match the state of the RF600E encoder's 4 digital inputs at the time of transmission. An additional output, which shows the status of the encoder battery, is also asserted.
The digital outputs may be configured as latching or momentary action.

Microcontroller

This project uses a Microchip PIC16F877A microcontroller. This component is a 40-Pin PDIP package, with 5 I/O ports, uses SPI and I^2C, has 256 bytes of internal EEPROM. (PIC16F87X) We chose to use the same microcontroller for all devices since this would make the programming and circuit design easier and more standardized. Our remote control requires the largest number of I/O pins, needing 6 input and 6 output pins. This microcontroller has more I/O pins than required, other features that we do not need, and the 40-pin PDIP package has a large footprint. We chose this microcontroller because the usage of it was being taught in our 5th semester COMP class, and we decided that this class would help us learn how to program and use this microcontroller in our design. We chose the PDIP package because it is of the form we can program using the available PICStart Plus programmer supplied by SIAST.

Relay

The G6BK-1114P relay by Omron Technology is a 5V dual coil latching relay that has a maximum current rating of 5A for 250VAC or 5A, 30VDC. The dual coil type is a set/reset coil; one set of pins sets the latch to allow the voltage to pass through to the device connected to the relay. The reset coil is used to disconnect the voltage from the device connected to the relay.

Electromagnet

The door lock mechanism uses two electromagnets powered with 5V and 2A.

Motor

The motor for the door opener operates with +/-5V and 250mA.

Antenna

¼ Wave Whip Stub Antenna made by R.F. Solutions and is tuned to the frequency of 433.92MHz.

Terminal Blocks

Used to connect power from the PCB to a device. Can handle 8A at 300VDC.

Programming

Language and Programmer Used

The C programming language was chosen for the code for this system. The CSCC C compiler was used to create files to be transferred into the microcontroller's memory. The MPLab Integrated Development Environment was used to transfer files into the microcontroller's memory through the PicStart Plus PIC Microcontroller Programmer made by Microchip.com, since the PIC16F877A microcontroller is fully supported.

Procedures and Descriptions

The code for each device can be found in Appendix X of the DyckCo Wireless Remotely Controlled Home System Technical Manual, after the report in this binding.

Remote Control Code

Main Procedure

Inside of an infinite loop, this procedure waits for a button press on the three inputs on PORTC coming from the buttons on the remote.

If button 1 is pressed, go to the Button1 procedure.

If button 2 is pressed, go to the Button2 procedure.

If button 3 is pressed, go to the Button3 procedure.

If one of the buttons were pressed, call the Receive procedure.

Button1 Procedure

Turn button 1 output high for 3 seconds to send the command to the Voltage Controller to trigger.

Button2 Procedure

Turn button 2 output high for 3 seconds to send the command to the Door Lock Controller to trigger.

Button3 Procedure

Turn button 3 output high for 3 seconds to send the command to the Door Motor Controller.

Receive Procedure

If the Voltage Controller responded, turn on the red LED for 2 seconds as the acknowledgement notification to the user.

If the Door Lock Controller responded, turn on the orange LED for 2 seconds as the acknowledgement notification to the user.

If the Door Motor Controller responded, turn on the green LED for 2 seconds as the acknowledgement notification to the user.

Wait for 5 seconds before getting new input from the user.

Voltage Controller Code

Main Procedure

Inside an infinite loop, wait for input on pin C0. If a signal is received, either set the relay by calling the Turnon procedure, or reset the relay by calling the Turnoff procedure, depending on the relay's current state.

Call the Transmit procedure.

Turnon Procedure

Turn the reset coil low, turn the set coil high. Turn on the red LED for 1 second as an acknowledgement notification.

Turn the set coil low.

Turnoff Procedure

Turn the set coil low, turn the reset coil high. Turn on the red LED for 1 second as an acknowledgement notification.

Turn the reset coil low.

Transmit Procedure

Transmit button 1 press for 5 seconds back to remote.

Wait 2 seconds extra until accepting new input.

Door Lock Controller

Main Procedure

Inside an infinite loop, wait for input on pin C1. If a signal is received, either set the relays by calling the Doorunlocked procedure to unlock the door, or reset the relays by calling the Doorlocked procedure to lock the door, depending on the lock's current state.

Call the Transmit procedure.

Doorunlocked Procedure

Turn the reset coil low; turn the set coil high on both electromagnets. Turn on the orange LED for 1 second as an acknowledgement notification.

Turn the set coils low.

Doorlocked Procedure

Turn the set coil low, turn the reset coil high on both electromagnets. Turn on the orange LED for 1 second as an acknowledgement notification.

Turn the reset coils low.

Transmit Procedure

Transmit button 2 press for 5 seconds back to remote.

Wait 2 seconds extra until accepting new input.

Door Motor Controller

Main Procedure

Inside an infinite loop, wait for input on pin C1. If a signal is received, either trigger one side of the H-Bridge by calling the turnopen procedure to open the door, or trigger the other side of the H-Bridge by calling the turnclose procedure to close the door, depending on the door's current state.

Call the Transmit procedure.

Turnopen Procedure

Turn high one pin input to the H-Bridge, and set the other pin low, to turn the motor counter-clockwise to open the door.

Turn on the red LED for 1 second as an acknowledgement notification.

Turn the pins low.

Turnclose Procedure

Turn high one pin input to the H-Bridge, and set the other pin low, to turn the motor clockwise to open the door.

Turn on the red LED for 1 second as an acknowledgement notification.

Turn the pins low.

Transmit Procedure

Transmit button 3 press for 5 seconds back to remote.

Wait 2 seconds extra until accepting new input.

See Appendix B for the flowcharts for the programming for each device.

See Appendix C for the programming code for each device.

Problems Encountered, Solutions Found

Serial Communication

Encryption is required for secure communication. With SPI, communication needs to be synchronized before messages can be transmitted or received between Microcontrollers. However signals cannot be decrypted without a signal first being synchronized. Whether a signal is a synchronizing signal or a data message cannot be determined until after a signal is synchronized and decrypted.

SPI is made for bi-directional communication in-circuit through "3-wire". "3-wire" is set up by three pins on each microcontroller. One pin is a clock pin used for synchronization and the other two pins are used for transmission or receiving. The receivers and transmitters connected to each microcontroller have only one pin for communication, so they cannot control which of the three pins on the PIC they are trying to communicate with. Therefore SPI will not work for our wireless system.

Our solution to not use the serial pins of the microcontroller is to use the RF600 series Keeloq ICs from R.F. Solutions. These ICs handle synchronization, encryption/decryption, and message determination internally, and communicate with the microcontroller directly. They are extremely user-friendly, but the encoders and decoders can only exchange messages consisting of which of the 4 buttons were pressed. It also communicates which encoder pushed the button.

Relay Voltage Divider

When the G6BK-1114P relay was connected into our Voltage Controller circuit directly to the output PORTD of the microcontroller, a voltage divider was created. The output pin of the microcontroller measured 1.7V instead of the 5V required to trigger the relay. This resulted in the inability of the set/reset coils of the relay to latch or de-latch the relay.

After consulting with Rob, our Networking instructor, we used a BJT as a buffer, as shown below in Figure M-2. This allowed us to get the full 5V from the microcontroller's output pins to trigger the set/reset coils of the relay so we could latch the relay and turn the light on in our system.

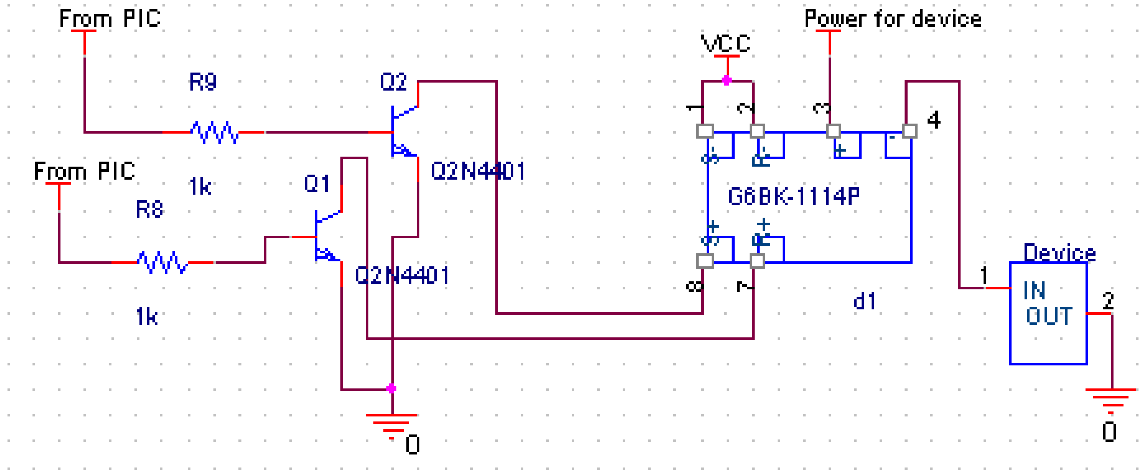

Figure M-2: BJT Buffer for Relay Schematic

Voltage Inverter for Motor

The motor for this system requires at least 5V to operate. The leads on the motor require opposite polarity inputs to the motor for it to turn. We tried a voltage inverter from Microchip.com, but it could handle 100mA while our motor required 250mA.

After consulting with Bob, our Control systems instructor, we decided to use an H-Bridge to provide the inputs to the motor, as shown below in Figure M-3. As one side of the H-Bridge is triggered, it produces +5V to one side of the motor and 0V to the other, turning the motor clockwise. As the other side of the H-Bridge is triggered, it produces +5V to that side, and 0V to the original side of the motor, turning the motor counter-clockwise.

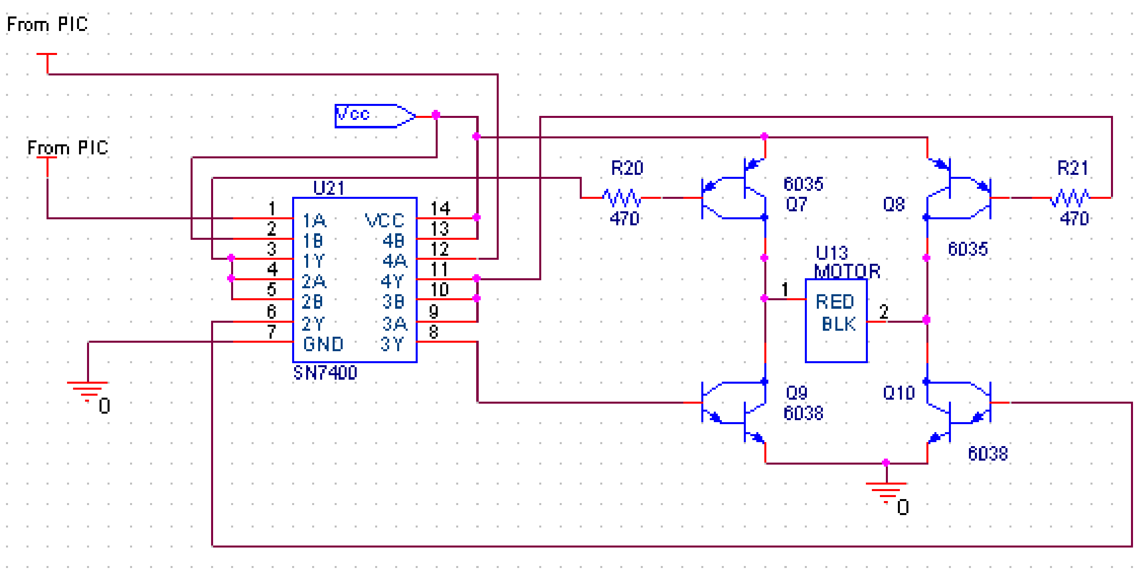

Figure M-3: H-Bridge Schematic for Motor

Battery Backup

The electromagnetic deadbolts for our door lock are unable to be opened during a power outage since they run off of line voltage. If there was a mechanical systems in place to allow the user to physically push the deadbolt open from the outside, this could be a security risk as anyone could unlock the door.

Instead of an insecure physical movement solution, a battery backup system would be better. If there was a manual key lock to the deadbolt that triggered a battery backup to apply power to the deadbolt, it would allow the user to unlock their door during an outage. This is not currently implemented in our system's design.

Future Implementation/Changes

This design was presented as a prototype on April 13, 2007. In the future, if this design was to be implemented, or made for distribution, the following changes should be made, and other additions could be added.

Surface Mount Technology (SMT)

SMT Antennas would provide an internal antenna, reducing the size of the remote, and creating no extra external components.

SMT Microcontrollers would create a smaller footprint, making the remote control smaller. These would require a SMT microcontroller programmer.

Pre-Made Electromagnetic Locks

Using electromagnetic locks for our system would make installation easier. They are stronger, smaller, and more efficient than the deadbolt we are currently using. They would have pre-configured specs readily available, so finding a lock to meet requirements would be easier.

Battery Backup

As mentioned previously, a battery backup circuit should be created to allow the user to have a key to trigger a battery to supply the electromagnetic deadbolt in case of a power outage.

Sleep Mode for Remote

This would allow the remote to have a longer battery life, making it cheaper, and easier to use for the user.

More Than 3 Devices, More Than One Remote

The RF600E can control up to four devices. Using multiple encoders in one remote could add multiple buttons to the remote, adding extra devices to the system.

The RF600D decoders can learn up to seven encoders, and with external EEPROM, they can learn up to 48 encoders. Therefore the RF600D decoder could respond to up to 48 remotes per decoder, enough for an extended family.

More Control over Command Signal

RF600D/E only communicate button pressed and low battery messages between them. Using the HCS series of Keeloq ICs from Microchip.com, complete control over encrypted messages can be achieved.

Having complete control over messages would allow extra data to be transmitted, such as the last time the door was opened, the current state of devices, synchronizing the clock time between devices, or anything else required.

Troubleshooting

- **Remote has no power:** Check battery.
- **Device will not respond:** Did you press the button? Did you press the correct button?
- **Voltage Controller has no power:** Check it is plugged into the wall and that there is power going to that outlet.
- **Door Lock Controller has no power:** Check it is plugged into the wall and that there is power going to that outlet.
- **Door Motor Controller has no power:** Check it is plugged into the wall and that there is power going to that outlet.
- **LED not lighting up:** Check if it is burnt out.
- **Receiver receives message, but decoder does not pass it to microcontroller:** Check if decoder is still programmed to listen to encoder sending command.
- **Cannot unlock deadbolt:** Check for power outage.
- **Cannot trigger relay set/reset coil:** Make sure the set coil gets +5V while the reset coil gets 0V, or vice versa. If the coil needing +5V is getting less than +5, check for a voltage divider, or check the BJT circuit.
- **Power supply not working:** Check the capacitors and the rectifier.
- **Power supply producing too much voltage:** Check the LM7805 voltage controller.
- **I smell burning:** Using caution, unplug the device immediately.

Expected Results

- PIC input pins require +5V to receive a logic high.
- PIC input pins require 0V to receive a logic low.
- PIC output pins should produce +5V at logic high.
- PIC output pins should produce 0V at logic low.
- AM-HRR30-433 should produce 1.7V at output pin when receiving signal.
- AM-HRR3-433 should produce 1.7V at output pin when receiving signal.
- RF600D output pins OP1-OP4 should produce +5V when triggered.
- RF600E input pins S0-S3 require logic low of 0V when triggered.
- H-Bridge should be provided +5 on one side and 0V on the other, if the 0V side receives more voltage, the H-Bridge will not be triggered, and may cause a short.

Technical Manual
Glossary

EEPROM Electrically Erasable Programmable Read-Only Memory, is a non-volatile storage chip used in computers and other devices to store small amounts of configuration data.

Keeloq RF solutions trademarked encryption technique.

RF Radio Frequency. The portion of the frequency spectrum in which electromagnetic waves can be generated by alternating current which is than fed to an antenna.

Sleep Mode The ability of the remote to reduce power consumption increasing battery life.

Wireless The use of some form of radio wave technology to transmit signals.

List of Acronyms

A – Amp, a measure of Current

AM – Amplitude Modulation

VAC – Voltage – Alternating Current

VDC – Voltage – Direct Current

Parts List

U1, U6, U11, U16 - PIC16C877A –Microchip 40-Pin Microcontroller

U2, U6, U12, U17 – RF600D – R.F. Solutions Keeloq Decoder

U3, U8, U13, U18 – RF600E – R.F. Solutions Keeloq Encoder

U4, U9 – AM-HRR30-433 – R.F. Solutions Miniature AM Super-Regenerative Receiver

U14, U19 – AM-HRR3-433 – R.F. Solutions AM Super Regenerative Receiver

U5, U10, U15, U20 – AM-RT4-433 – R.F. Solutions AM Hybrid Transmitter

d1, d2, d3 – G6BK-1114P – Omron Electronics Relay

Q1, Q2, Q3, Q4, Q5, Q6 – 2N4401 – National Semiconductors BJT

R1, R4, R7, R8, R9, R13, R14, R15, R16, R17 - 1kΩ Resistor

R2, R10, R18 - 15kΩResistor

R3, R11, R19 - 22kΩ Resistor

R20, R21 - 470Ω Resistor

XTL1, XTL2, XTL3, XTL4 - 10 MHz Resonator

C1, C2, C4, C5, C7, C8, C10, C11 - 33pF Capacitors

C3, C6, C9, C12 - 10nF Capacitors

SW1, SW2, SW3 – Switches

D1, D6 - LED Red

D2, D8 - LED Orange

D3, D10 - LED Green

D4, D5, D7, D9 - LED Yellow

Light – 120V Line Voltage Plug-in Light

Motor – 5V DC Motor

Electromagnet1, Electromagnet2 - Electromagnets

Q7, Q8 - 6035 – PNP Darlington Pair

Q9, Q10 - 6038 – NPN Darlington Pair

U21 - SN7400 - Quad NAND IC

Battery Holder – Keystone 1026

Suppliers

Motor: George at Northcote Sales, Regina, Saskatchewan

Digikey.ca:

- Microchip.com: PIC16F877A Microcontroller
- R.F. Solutions: RF600E/D, AM-RT4-433, AM-HRR30-433, AM-HRR3-433
- Linx Technologies: ¼ Wave Whip Stub Antenna
- Omron Electronics: G6BK-1114P Dual Coil Latching Relay
- Keystone: Dual Cell Holder 1026 – 2 cell 20mm.

B&E Regina:

- Terminal Blocks

SIAST:

- PCB: 2oz copper, single and dual sided.
- Cambions
- 2N4401
- LM7805
- Transformer
- 6035
- 6038
- SN7400
- Capacitors, Resistors, Resonators, LEDs, Switches.

Bibliography

Linx Technologies. "ANT-433-PW-LP Data Sheet". LP Series - Reduced height straight screw mount 1/4 wave monopole antenna... <http://www.antennafactor.com/documents/ANT-433-PW-LP_Data_Sheet.pdf>. January 2007.

Microchip.com. "PIC16F87X Data Sheet". PIC16F87X Data Sheet. <http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf > July 2006.

Omron Technologies. "G6B Series". G6B Series. <http://oeiwcsnts1.omron.com/pdfcatal.nsf/PDFLookUpByUniqueID/3AD1CDB9F74BA46686256FC7005CEA7A/$FILE/D20G6B0305.pdf> January 2007

Pailsey, R. "H-Bridge Motor Control Circuits." H-Bridge Motor Control Circuits. <http://home.cogeco.ca/~rpaisley4/HBridge.html> April 2007.

R.F Solutions. "DS013: AM-RT4-433". R.F. Solutions Ltd AM Transmitter (4 Pin) <http://www.rfsolutions.co.uk/acatalog/DS013-6_AM-RTx.pdf> July 2006.

R.F Solutions. "DS014: AM-HRR30-433". R.F. Solutions Ltd AM Miniature Super-Regen Receivers <http://www.rfsolutions.co.uk/acatalog/DS014-1.pdf> July 2006.

R.F Solutions. "DS016: AM-HRR3-433". R.F. Solutions Ltd AM Super-Regen Receivers <http://www.rfsolutions.co.uk/acatalog/DS016-13_AM-HRR.pdf> July 2006.

R.F Solutions. "DS600: RF600E and RF600D". R.F. Solutions Ltd Keeloq Encryption Chipset. <http://www.rfsolutions.co.uk/acatalog/DS600-10_RF600.pdf> July 2006.

Traylor. "Using the BJT as Saturated Switch." Using the BJT as Saturated Switch. <http://web.engr.oregonstate.edu/~traylor/ecell2/lectures/bjt_sat_switch.pdf> February 2007.

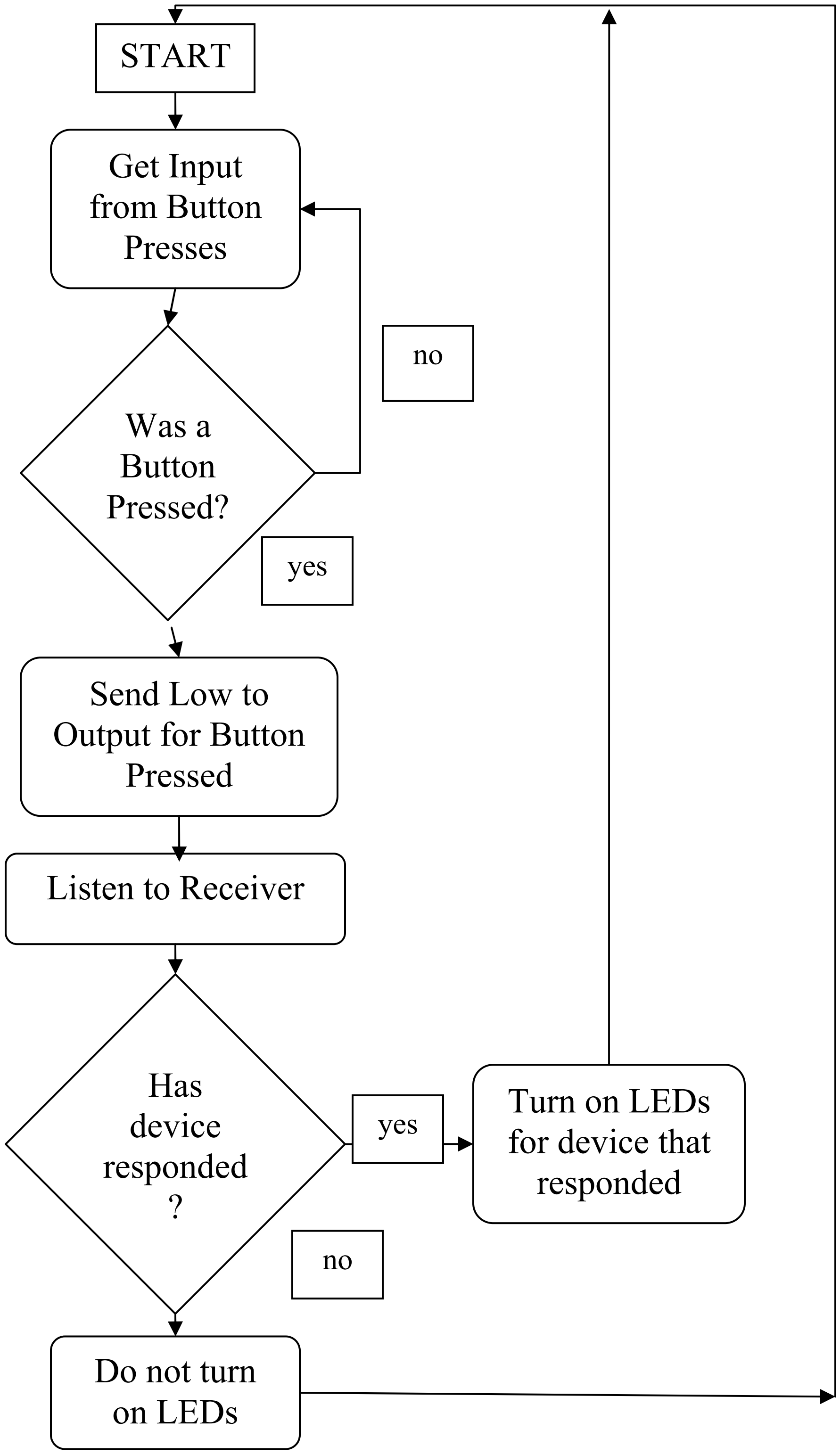

Figure M-A-1 – DyckCo Remote Control Flowchart

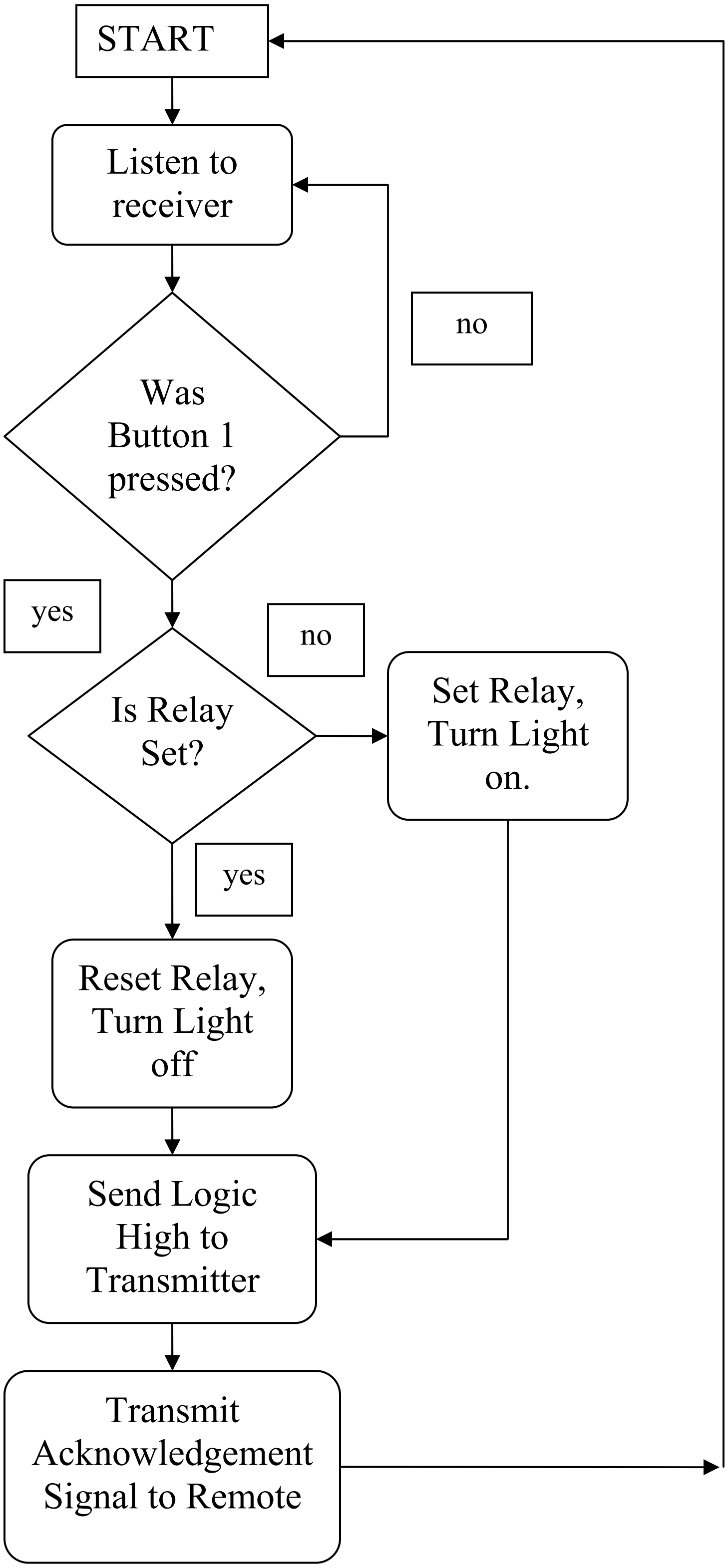

Figure M-A-2 – DyckCo Voltage Controller Flow Chart

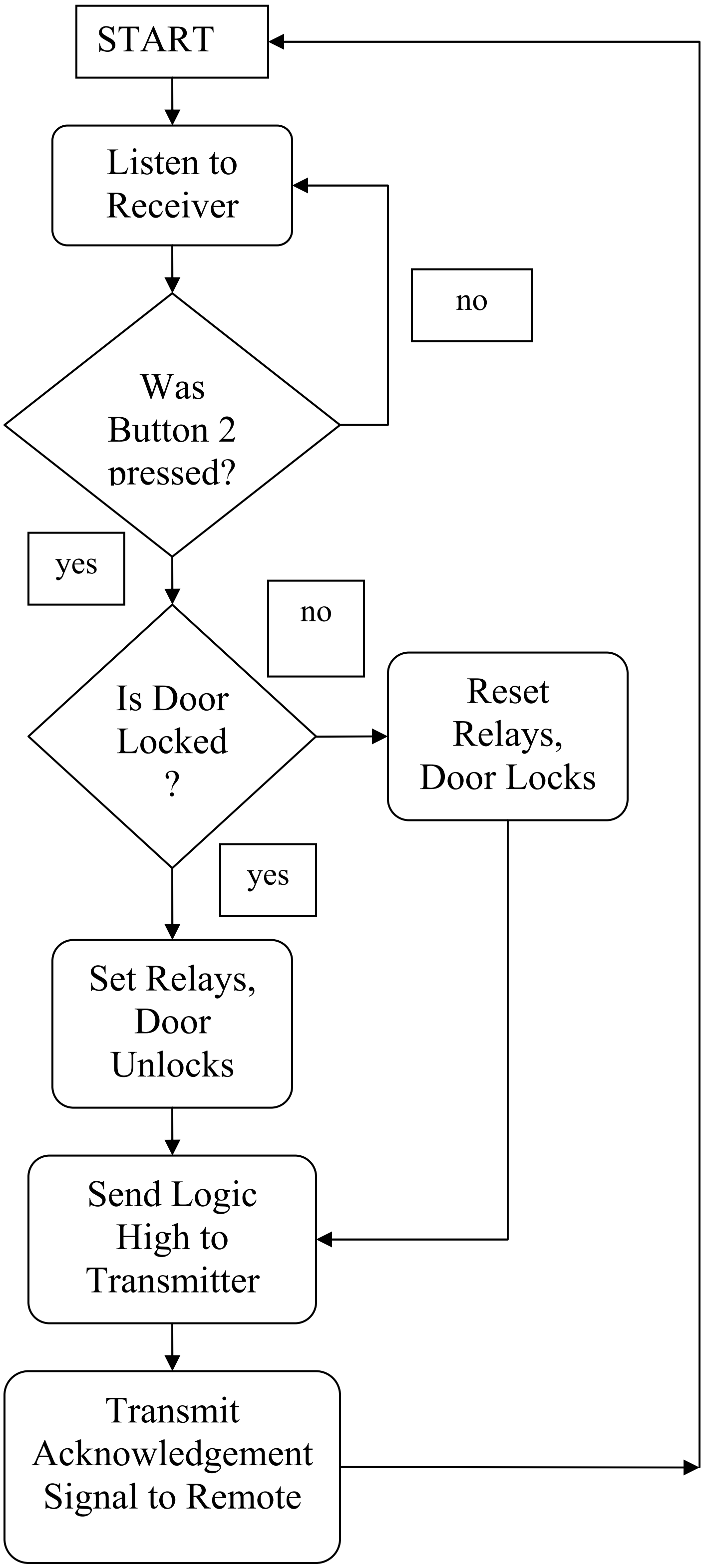

Figure M-A-3– DyckCo Door Lock Controller Flowchart

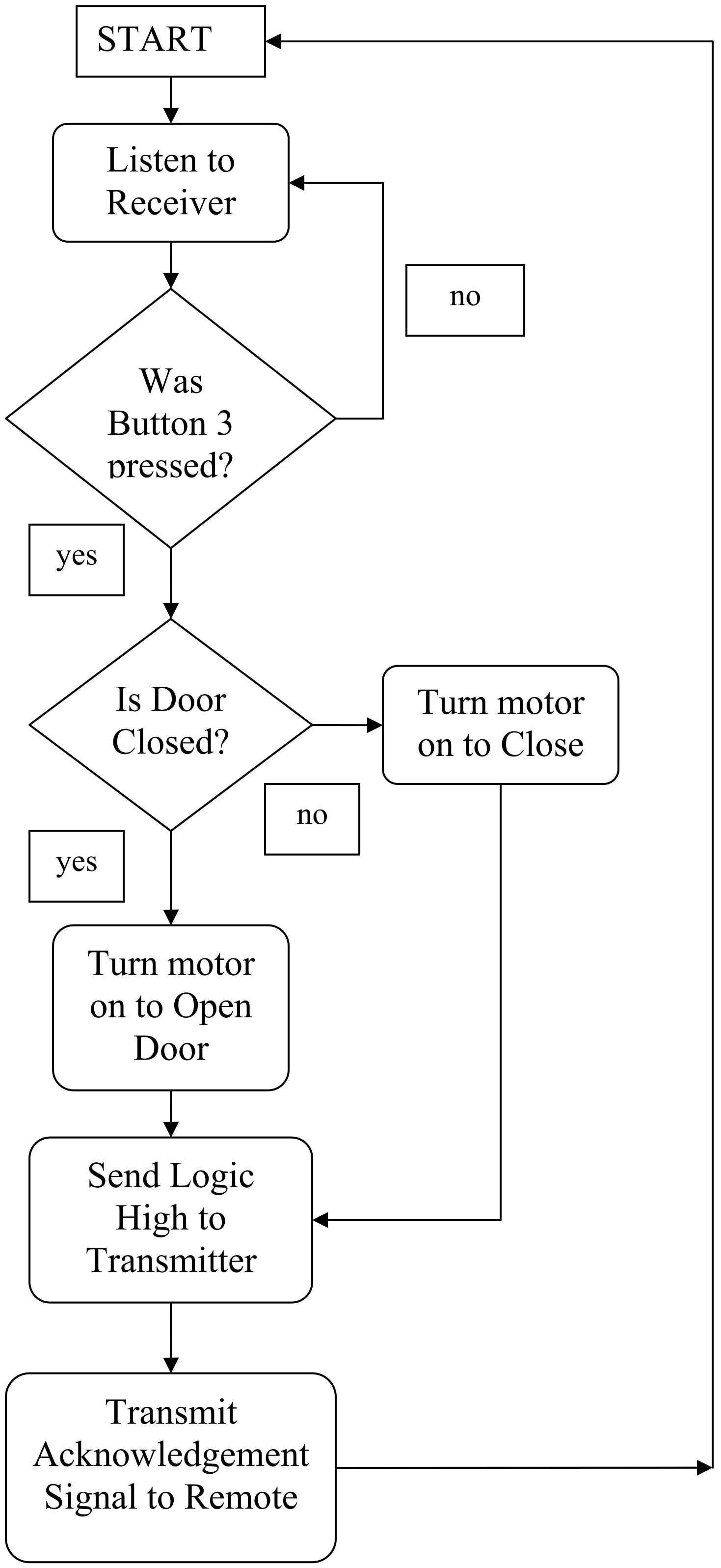

Figure M-A-4 – DyckCo Door Motor Controller Flowchart

APPENDIX B – Schematics of Devices

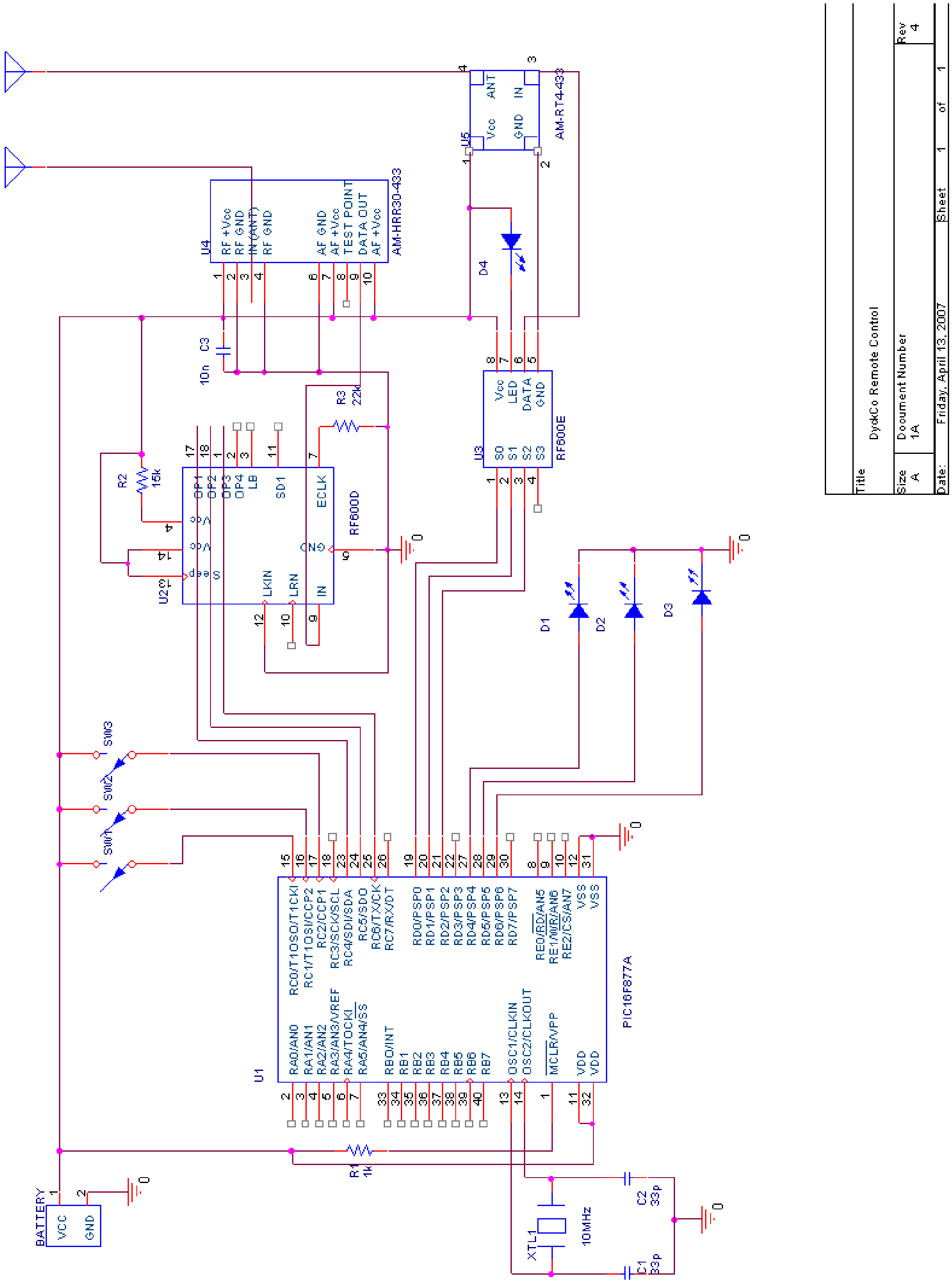

Figure M-B-1 – DyckCo Remote Control Schematic

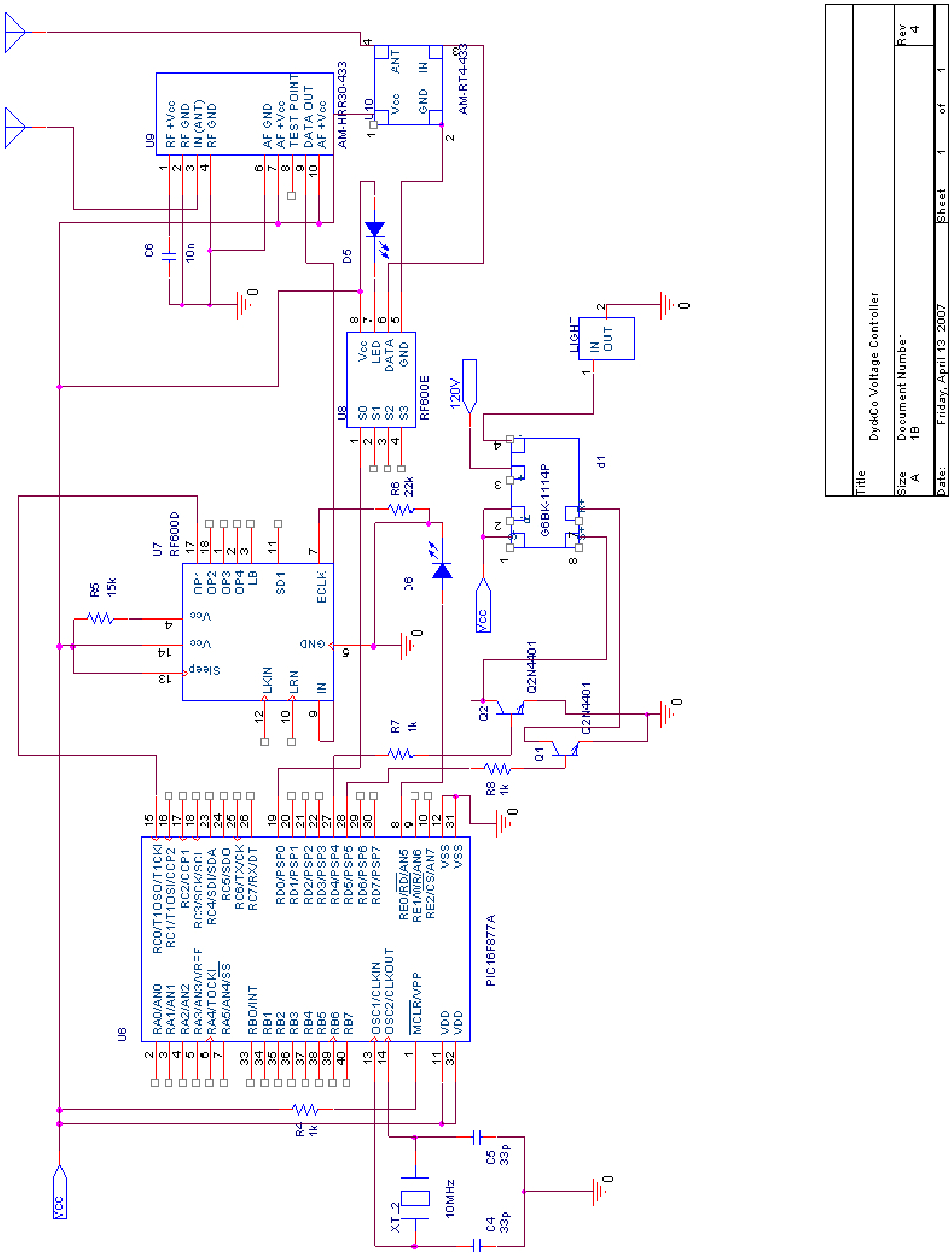

Figure M-B-2 – DyckCo Voltage Controller Schematic

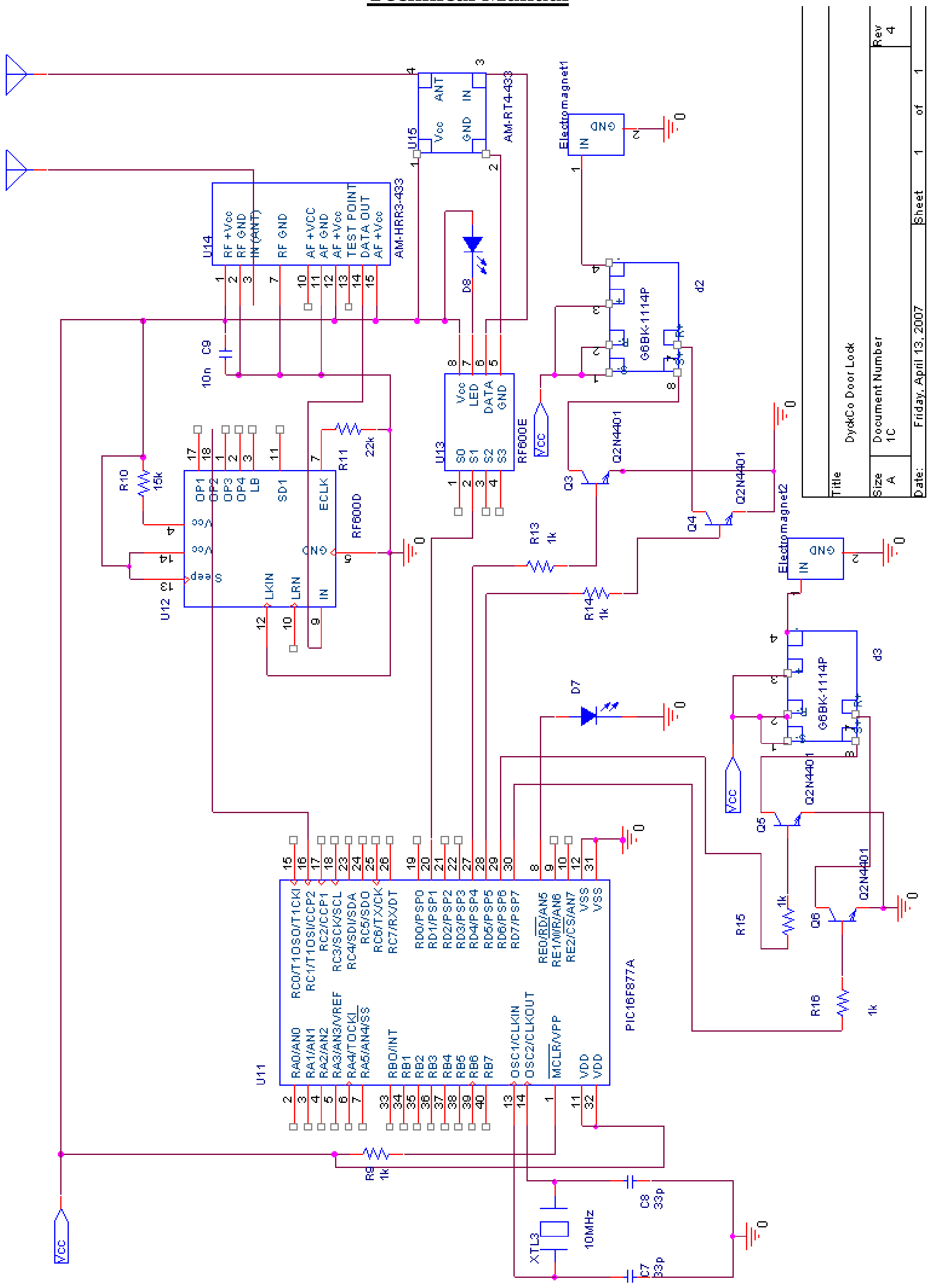

Figure M-B-3 – DyckCo Door Lock Controller Schematic

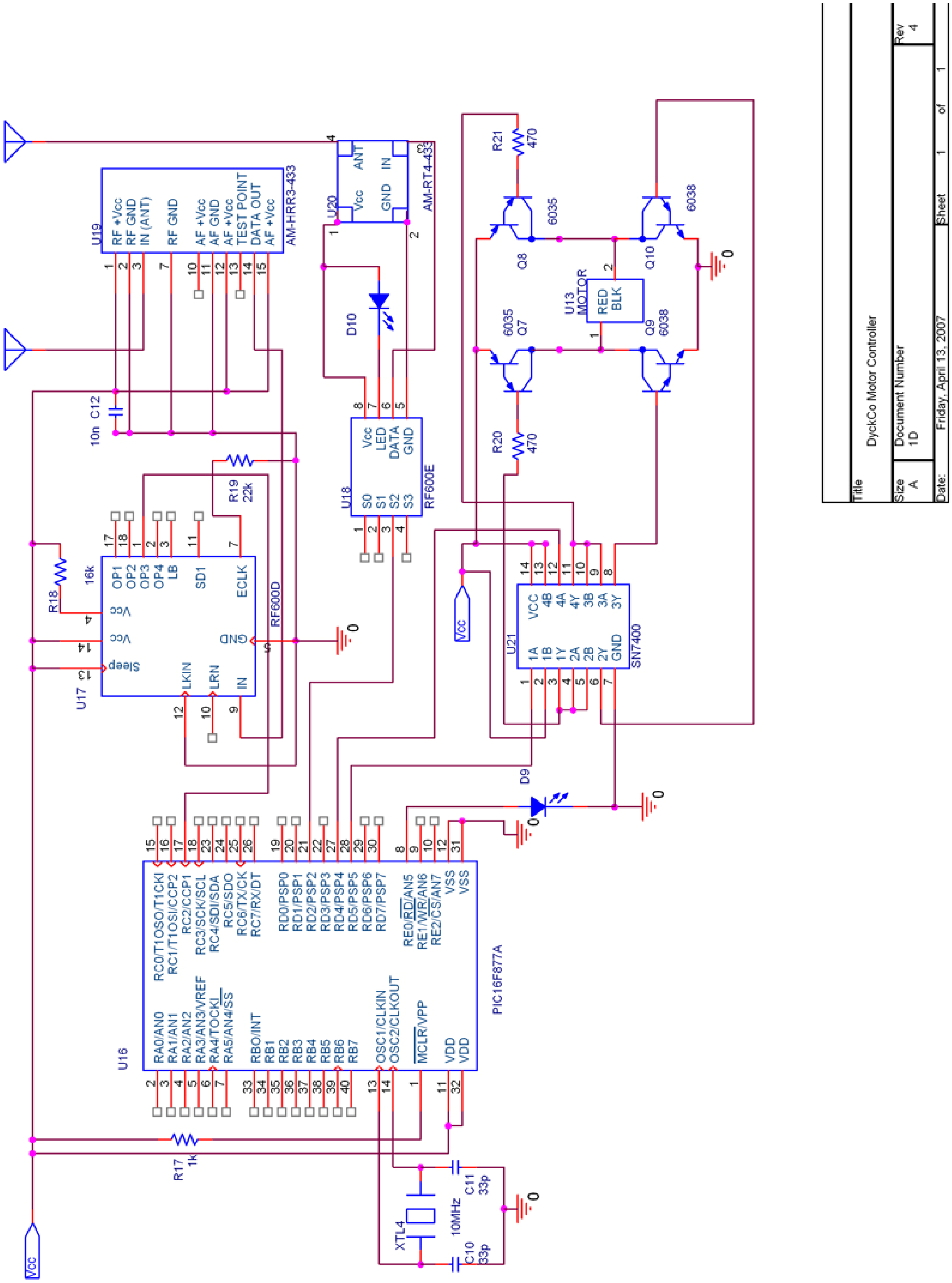

Figure M-B-4 – DyckCo Door Motor Controller Schematic

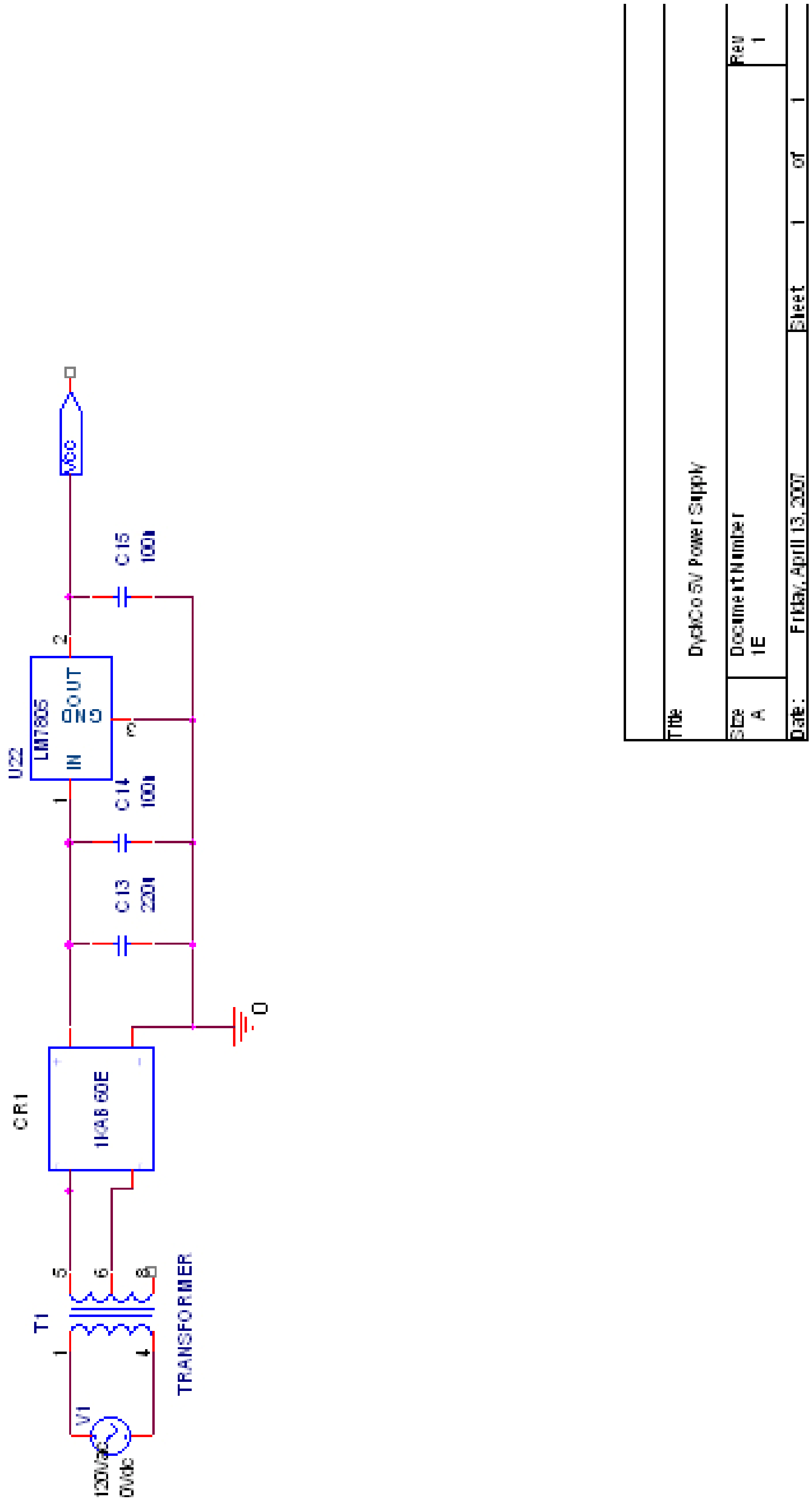

Figure M-B-5 – DyckCo 5V Power Supply Schematic

APPENDIX C - Program Code

DyckCo Remote Control Code

```
// Trevor Cook
// April 11, 2007
// Final Project

#include <16F877A.h>
#use delay(clock=10000000)
#fuses HS, NOWDT

int did;

void button1() // If Button1 Pressed (Voltage Controller)
{
        output_high(PIN_D0);        // PORTD.0 pin 19
        delay_ms(3000);
        output_low(PIN_D0);         // PORTD.0 pin 19
        did = 1;
}
void button2() // If Button2 Pressed (Door Lock)
{
        output_high(PIN_D1);        // PORTD.1 pin 20
        delay_ms(3000);
        output_low(PIN_D1);         // PORTD.1 pin 20
        did = 1;
}
void button3() // If Button3 Pressed (Door Motor)
{
        output_high(PIN_D2);        // PORTD.2 pin 21
        delay_ms(3000);
        output_low(PIN_D2);         // PORTD.2 pin 21
        did = 1;
}
void Receive()
{
        delay_ms(1500);                     // Wait for 1.5 sec turn on time

        if (!input(PIN_C4))                 // If Button1 from RX
        {
                output_high(PIN_D4);        // PORTD.4 pin 27
                delay_ms(2000);             // Turn on LED1 for 2 sec
                output_low(PIN_D4);
        }
        else if (!input(PIN_C5))            // If Button2 from RX
        {
```

```
            output_high(PIN_D5);        // PORTD.5 pin 28
            delay_ms(2000);             // Turn on LED2 for 2 sec
            output_low(PIN_D5);
        }
        else if (!input(PIN_C6))        // If Button3 from RX
        {
            output_high(PIN_D6);        // PORTD.6 pin 29
            delay_ms(2000);             // Turn on LED3 for 2 sec
            output_low(PIN_D6);
        }
        delay_ms(5000);                 // Wait 5 sec
        output_d(00000000);             // Set PORTD off
        did = 0;                        // Reset value
}

void main()
{
        set_tris_c(0xff);
        set_tris_d(0x00);
        did = 0;

        while(1)
        {
            if (input(PIN_C0) == 1)
                        // if PORTC.0 Pin 15 = High, Button 1 was pressed
            {
                button1();
            }
            if (input(PIN_C1) == 1)
                        // if PORTC.1 Pin 16 = High, Button 2 was pressed
            {
                button2();
            }
            if (input(PIN_C2) == 1)
                        // if PORTC.2 Pin 17 = High, Button 3 was pressed
            {
                button3();
            }
            if (did != 0)
            {
                receive();      // Call Receiving Procedure
                did = 0;        // Reset value just in case.
            }
        }
}       // End of Main Procedure and End of Program
```

DyckCo Voltage Controller Code

```
// Trevor Cook
// April 12, 2007
// Final Project

#include <16F877A.h>
#use delay(clock=10000000)
#fuses HS, NOWDT

int open, did;

void turnon()
{
        output_low(PIN_D4);         // PORTD.4 pin 27 Turn off Reset on RELAY
        output_high(PIN_D5);        // PORTD.5 pin 28 Turn on Set on RELAY
        output_high(PIN_E0);        // PORTE.0 pin 8 Turn on LED
        delay_ms(1000);             // Wait 1 second
        output_low(PIN_D5);         // Turn off Set command on RELAY
        open=1;                     // Set open = 1, Device is turned on
        did = 1;                    // Set did = 1, turned on device
        output_low(PIN_E0);         // PORTE.0 pin 8 Turn off LED
}

void turnoff()
{
        output_low(PIN_D5);         // PORTD.5 pin 28 Turn off Set on RELAY
        output_high(PIN_D4);        // PORTD.4 pin 27 Turn on Reset on RELAY
        output_high(PIN_E0);        // PORTE.0 pin 8 Turn on LED
        delay_ms(1000);             // Wait 1 second
        output_low(PIN_D4);         // Turn off Reset command on RELAY
        open=0;                     // Set open = 0, Device is turned off
        did = 0;                    // Set did = 0, turned off device
        output_low(PIN_E0);         // PORTE.0 pin 8 Turn off LED
}

void transmit()
{
        output_high(PIN_D0);        // PORTD.0 pin 19 Turn on Button 1 Press
        delay_ms(5000);             // Wait 5 seconds
        output_low(PIN_D0);         // PORTD.0 pin 19 Turn off Button 1 Press
        did = 0;                    // Set did = 0, transmit successful
        delay_ms(2000);             // wait to read next button press
                                    // to make sure we've stopped transmitting
}
```

```
void main()
{
	set_tris_c(0xff);
	set_tris_d(0x00);
	set_tris_e(0x00);
	open = 0;
	did = 0;

	while(1)
	{
		if (!input(PIN_C0))		// PORTC.0 pin 15, IF Input Low
		{
			if (open == 0)		// If relay is open, goto procedure to close it
			{
				turnon();	// Call procedure turnon
			}
			else if (open == 1 && did == 0)
					// If relay is closed and we didn't do anything yet
			{		// then call procedure to open it
				turnoff();	// Call procedure turnoff
			}
			delay_ms(2000);		// Wait for remote to start listening
			transmit();		// Transmit Acknowledgement signal
		}
		if (!input(PIN_C1))		// ignore everything else
		{
			output_low(PIN_E0);
			output_low(PIN_D0);
		}
		if (!input(PIN_C2))		// ignore everything else
		{
			output_low(PIN_E0);
			output_low(PIN_D0);
		}
		else {				// ignore everything else
			output_low(PIN_E0);
			output_low(PIN_D0);
			}
	}
}	// End of Main Procedure and End of Program
```

DyckCo Door Lock Controller Code

```
// March 28, 2007
// Trevor Cook
// Final Project

#include <16F877A.h>
#use delay(clock=10000000)
#fuses HS, NOWDT

int open, did;

void doorunlocked()
{                               // Turn on Electromagents
        output_high(PIN_D4);    // PORTD.4 pin 27 Turn on Set on RELAY1
        output_low(PIN_D5);     // PORTD.5 pin 28 Turn off Reset on RELAY1
        output_high(PIN_D7);    // PORTD.7 pin 30 Turn on Set on RELAY2
        output_low(PIN_D6);     // PORTD.6 pin 29 Turn off Reset on RELAY2

        output_high(PIN_E0);    // PORTE.0 pin 8 Turn on LED

        delay_ms(1000);         // Wait 1 second

        output_low(PIN_D4);     // Turn off Set command on RELAY1
        output_low(PIN_D7);     // Turn off Set on RELAY2

        open=1;                 // Set open = 1, Door is Unlocked
        did = 1;                // Set did = 1, Unlocked Door
        output_low(PIN_E0);     // PORTE.0 pin 8 Turn off LED
}

void doorlocked()
{                               // Turn off Electromagnets

        output_high(PIN_D5);    // PORTD.5 pin 28 Turn on Reset on RELAY1
        output_low(PIN_D4);     // PORTD.4 pin 27 Turn off Set on RELAY1
        output_high(PIN_D6);    // PORTD.6 pin 29 Turn on Reset on RELAY2
        output_low(PIN_D7);     // PORTD.7 pin 30 Turn off Set on RELAY2

        output_high(PIN_E0);    // PORTE.0 pin 8 Turn on LED

        delay_ms(1000);         // Wait 1 second

        output_low(PIN_D5);     // Turn off Reset command on RELAY1
        output_low(PIN_D6);     // Turn off Reset command on RELAY2
```

```
        open=0;                         // Set open = 0, Door is Locked
        did = 0;                        // Set did = 0, Locked Door
        output_low(PIN_E0);             // PORTE.0 pin 8 Turn off LED
}

void transmit()
{
        output_high(PIN_D1);            // PORTD.1 pin 20 Turn on Button 2 Press
        delay_ms(5000);                 // Wait 5 seconds

        output_low(PIN_D1);             // PORTD.1 pin 20 Turn off Button 2 Press
        output_D(00000000);             // Clear PORTD

        did = 0;                        // Set did = 0, transmit successful
        delay_ms(2000);                 // wait to read next button press
                                        // to make sure we've stopped transmitting
}

void main()
{
        set_tris_c(0xff);
        set_tris_d(0x00);
        set_tris_e(0x00);
        open = 0;
        did = 0;

        while(1)
        {
                if (!input(PIN_C1))                     // PORTC.1 pin 16, If Input Low
                {
                        if (open == 0)                  // If door is locked, open it
                        {
                                doorunlocked();         // Call procedure doorunlocked
                        }
                        else if (open == 1 && did == 0)
                                        // If door is unlocked and we didn't do anything yet
                        {               // then call procedure to lock it
                                doorlocked();           // Call procedure doorlocked
                        }
                        delay_ms(2000);                 // Wait for remote to start listening
                        transmit();                     // Transmit Acknowledgement signal
                }
        }
}       // End of Main Procedure and End of Program
```

DyckCo Door Motor Controller Code

```
// March 28, 2007
// Trevor Cook
// Final Project

#include <16F877A.h>
#use delay(clock=10000000)
#fuses HS, NOWDT

int open, did;

void turnopen()                     // Open Door (Turn Motor Counter-clockwise)
{
        output_low(PIN_D5);         // PORTD.5 pin 28
        output_high(PIN_D4);        // PORTD.4 pin 27 Turn on Motor Counter-
Clockwise
        output_high(PIN_E0);        // PORTE.0 pin 8 Turn on LED

        delay_ms(1500);             // Wait 1 second

        output_low(PIN_D4);         // Turn off Motor

        open = 1;                   // Set open = 1, Door is Opened
        did = 1;                    // Set did = 1, Opened Door
        output_low(PIN_E0);         // PORTE.0 pin 8 Turn off LED

}

void turnclose()                    // Close Door (Turn Motor Clockwise)
{
        output_low(PIN_D4);         // PORTD.4 pin 27
        output_high(PIN_D5);        // PORTD.5 pin 28 Turn on Motor - Clockwise
        output_high(PIN_E0);        // PORTE.0 pin 8 Turn on LED

        delay_ms(1500);             // Wait 1 second

        output_low(PIN_D5);         // Turn off Motor

        open = 0;                   // Set open = 0, Door is Closed
        did = 0;                    // Set did = 0, Closed Door
        output_low(PIN_E0);         // PORTE.0 pin 8 Turn off LED
}
```

```
void transmit()                     // Transmit Acknowledgement Signal
{
        output_high(PIN_D2);        // PORTD.2 pin 21 Turn on Button 3 Press
        delay_ms(5000);             // Wait 5 seconds

        output_low(PIN_D2);         // PORTD.2 pin 21 Turn off Button 3 Press
        output_D(00000000);         // Clear PORTD

        did = 0;                    // Set did = 0, transmit successful
        delay_ms(2000);             // wait to read next button press
                                    // to make sure we've stopped transmitting
}

void main()
{
        set_tris_c(0xff);                       // PORTC is Inputs
        set_tris_d(0x00);                       // PORTD is Outputs
        set_tris_e(0x00);                       // PORTE is Outputs
        open = 0;                               // Initialize to Zero
        did = 0;                                // Initialize to Zero

        while(1)
        {
                if (!input(PIN_C2))             // PORTC.1 pin 17, If Input Low
                {

                        if (open == 0)          // If door is closed, open it
                        {                       // Start of If Statement
                                turnopen();     // Call procedure turnopen
                        }                       // End of If Statement

                        else if (open == 1 && did == 0)
                                    // If door is open and we didn't do anything yet
                        {           // then call procedure to close it
                                turnclose();    // Call procedure turnoff
                        }                       // End of If Statement

                        delay_ms(2000);         // Wait for remote to start listening
                        transmit();             // Transmit Acknowledgement signal
                }                               // End of If Statement

        }                                       // End of Infinite While Loop

}       // End of Main Procedure and End of Program
```