Parallax 2010 RF Design Contest:

Experimental Robotics Platform

Stephen Emery
StephenAEmery3@aol.com
Project ID: RF104322

Table of Contents:

Project Number:

RF104322

Project Description:

I have designed and built an experimental robotics platform using the drive system of an electric scooter. It has several important and useful features, including remote control, emergency stops, and shutdown circuitry. In this report I will provide a clear description of these features so that others may incorporate them into their projects.

Ever since elementary school, I have wanted to build a robot. Sure, I've built a few from kits, but I wanted to build one from scratch. One I could tweak, modify, and redesign. A platform I could use to explore concepts such as autonomy and telepresence, one large enough to experiment with any sensors or actuators I could think of. I've always had so many ideas for projects that building a large robot always got pushed off to the side because of the time and the cost required. But the day I found an old Shoprider scooter at a garage sale, I knew the time had come.

After restoring the scooter to working order, I began working out how to interface it with a microcontroller. I soon discovered that the most expedient method was to emulate the analog signals of the joystick and throttle control on the motor controller. This was simple enough, but I was not familiar with using one joystick to control two wheels, so I wanted to get a feel for the platform's range of motion and control scheme. I needed a way to control the robot from a distance so I could do this without tripping over it. I decided that the most practical approach would be to relay commands from a remotely located joystick to the motor controller. I chose the 433 MHz RF Transceivers for their ease of integration: the serial interface makes sending and receiving data straightforward, and the 0.1" spaced header lend themselves to the prototyping phase.

Adding remote control to the project presented a design challenge: what does the robot do when it does not have a signal? The answer should be "nothing". Thus, adding and out-of-range halt feature to the code was critical. Additionally, I decided that the robot should also have a user-activated E-stop that was separate from the main control circuitry, so that if there was a problem with the software, the robot could still be stopped. A keyless entry system for an automobile was used for this purpose. The E-stop is designed so that one button stops the robot, then that button must be pressed again to reset the robot, and then another button must be pressed to re-enable the robot. The reason for this is so that pressing the stop button repeatedly, as one may do in case of panic, will not reactivate the robot. There is no enable button on the robot so that the operator must be in possession of the E-stop fob in order to use the robot.

The startup and shutdown sequences of the robot are very important, because the motor controller will drop into an unresponsive error state if it receives out-of-bounds inputs. To remedy this problem, the Stamp was given control of its own voltage supply so it could manage the startup and shutdown sequences.

Schematics:

Figure 1 shows the schematic for the remote control. The user inputs are a joystick (R5) and a potentiometer (R4). The Stamp (U1) reads these inputs and the transmits them to the robot using the 433 MHz Transceiver (U2) (see Source Code for details). The Stamp is mounted on the Board of Education (BOE), so several connections have been omitted for clarity, as they have been documented elsewhere.

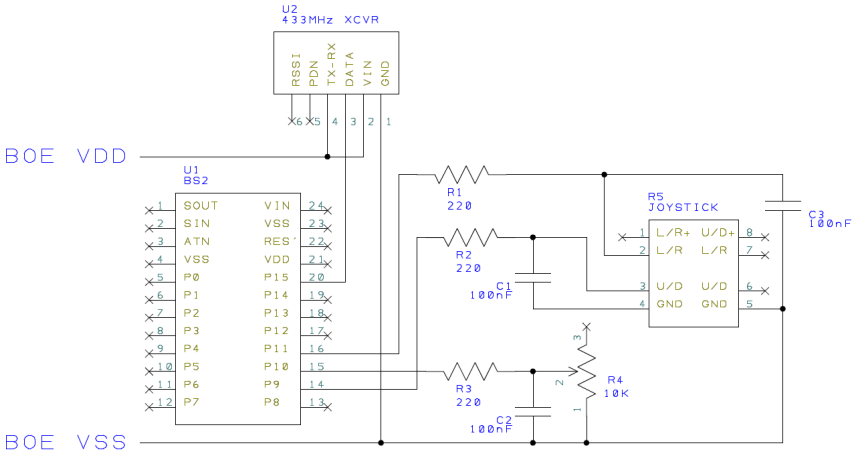

FIGURE 1: Remote Control

Figure 2 shows the schematic for the robot. The communication and control subsystems will be discussed in the Block Diagram section below, but the power subsystem will be discussed here. The robot runs on 24V, supplied by two 12V, 36Ah SLA batteries. 12V is taken from the battery connected to ground and used to power two cold-cathode fluorescent lamps (not shown for clarity) and the robot communication and control circuitry. When the user presses the power button (SW1), Q1 gets turned on, supplying 12V to the 5V regulator (U1). The Stamp (U2) has until the user releases the power button to turn on Q2, which keeps Q1 turned on. After the Stamp makes some other initializations (see Source Code for details), it turns on the motor controller. When the user pushes the power button again, P14 on the Stamp will go low. This tells the Stamp to turn off the motor controller, and then turn off Q2, which turns off Q1 and the rest of the robot with it. There are two slight drawbacks to this circuit. First, the power button must be held down while downloading programs, since it is the BS2 that latches the power. Second, a side effect of the power button being the same button that gets read by the Stamp is that current leaks through the diode (D2) while the system is off. It's only around 200nA, which is not a problem for these large batteries. However, separating the "on" and "request shutdown" functions of SW1 by replacing it with a DPDT would eliminate this problem.

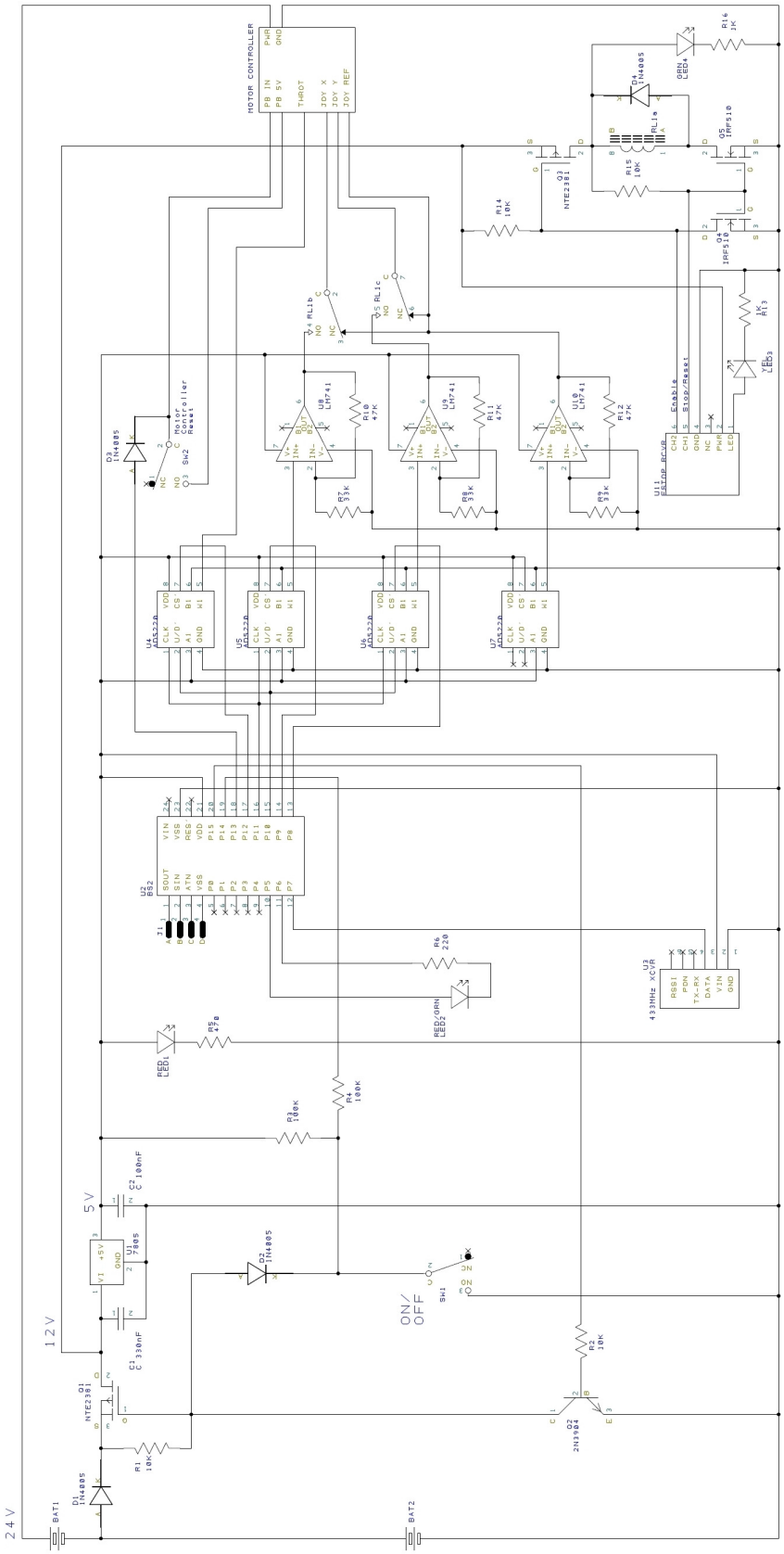

FIGURE 2: Robot Electronics

Figure 3 shows the programming adapter for the robot. It is very convenient because it has the two capacitors required to program the BASIC Stamp, so the only component needed on a board is the 4 pin header. Also, the capacitors can be by switched out of the circuit, which has uses in other applications.

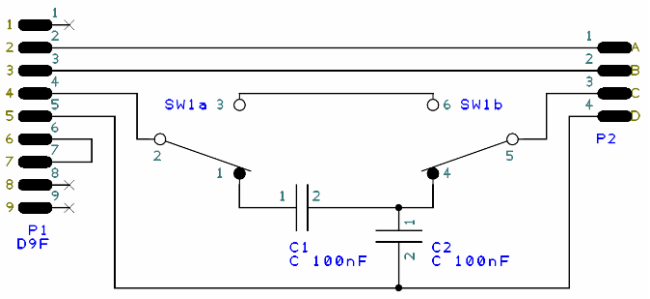

FIGURE 3: Programming Adapter

Block Diagram:

This section describes the communication and control of the system. Figure 4 shows the block diagram, and Figures 6-8 (Pictures section) show where these blocks are located in the system. The robot receives data for the remote control using a 433 MHz Transceiver (6). The Stamp (7) reads the data from transceiver, checks it for errors, and sets the DAC (8) accordingly (see Source Code for details). The DAC is composed of 4 digital potentiometers which simulate the original throttle and joystick inputs to the motor controller. One is used directly to control the throttle, and outputs of the other three are scaled up using op-amps to set the X, Y, and reference joystick inputs (the digital potentiometer used to set the joystick reference is not controlled by the Stamp, it stays in its initial center position).

The E-stop transmitter (1) and receiver (9) are used with a relay (10) to disable the robot by setting both the X and Y inputs of the motor controller to the reference voltage. To the motor controller, it appears that the user has let go of the joystick. The ideal E-Stop would be to turn the whole system off, but some online documentation for the motor controller states that it can be damaged by turning it off while the wheels are moving. The scheme used on the robot is a reasonable compromise, because it is unlikely that the motor controller itself will be the cause of a problem, since it has undergone far more rigorous engineering and testing than the rest of the robot, which is being built for the first time. However, future designs should include a motor controller that can be shut off while the robot is moving.

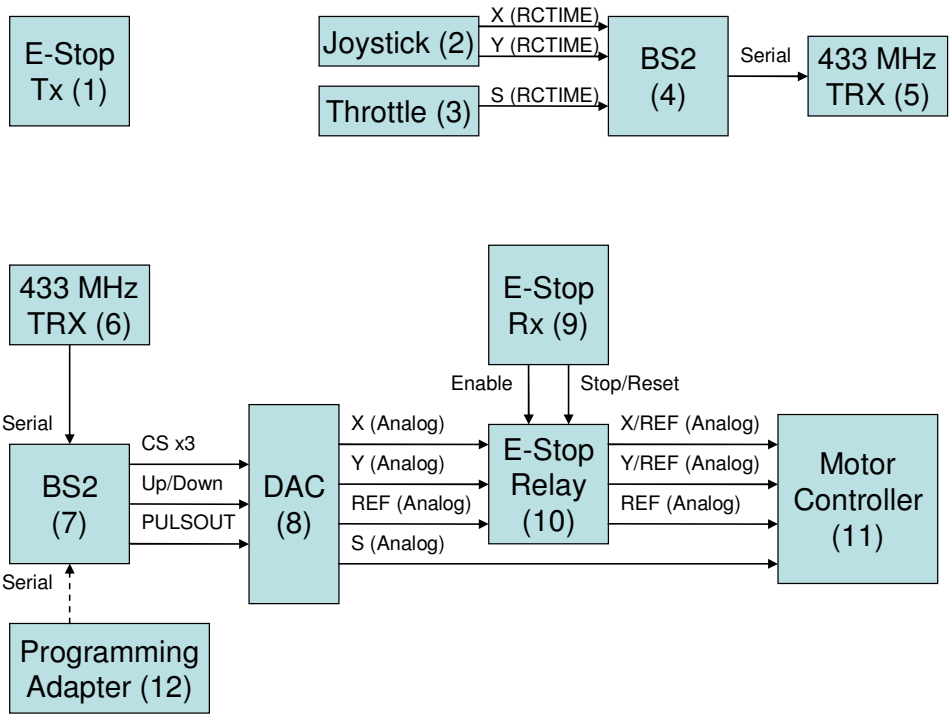

FIGURE 4: System Block Diagram

Source Code:

The code for the remote control uses the RCTIME function to read the positions of the joystick an the throttle potentiometer. It converts this data into the format used to set the DAC on the robot, and then transmits it serially with a 2-byte start-of-message identifier.

```
'=====================================================================================
'Robot_Xmt.bs2
'Stephen Emery
'05/31/2010
'=====================================================================================

' {$STAMP BS2}
' {$PBASIC 2.5}

'PINS=================================================================================

Xmt   PIN 15  '433 MHz Transceiver

XPot  PIN 11  'L/R Joystick Pot
YPot  PIN 9   'U/D Joystick Pot
SPot  PIN 10  'Throttle Pot (s is for speed)

'CONSTANTS============================================================================

B2400 CON 396   '2400 Baud, 8-Bit, No Parity
```

```
XYMax CON 76    'Maximum value for the joystick outputs of the robot's DAC
XYCen CON 64    'Center value for the joystick outputs of the robot's DAC
XYMin CON 50    'Minimum value for the joystick outputs of the robot's DAC
SMax  CON 127   'Maximum value for the throttle output of the robot's DAC

'VARIABLES=========================================================================

xTime  VAR Word
yTime  VAR Word
sTime  VAR Word

x  VAR Byte
y  VAR Byte
s  VAR Byte

'MAIN ROUTINE======================================================================

DO

  'Get potentiometer positions

  HIGH XPot
  HIGH YPot
  HIGH SPot

  PAUSE 100

  RCTIME XPot, 1, xTime
  RCTIME YPot, 1, yTime
  RCTIME SPot, 1, sTime

  'Convert position data to the format used by the robot's DAC

  IF (xTime > 378) THEN
    x = xTime*11/215 + 45
  ELSEIF (xTime < 338) THEN
    x = xTime*13/279 + 49
  ELSE
    x = XYCen
  ENDIF

  IF (yTime > 370) THEN
    y = yTime*11/277 + 50
  ELSEIF (yTime < 296) THEN
    y = yTime*13/279 + 50
  ELSE
    y = XYCen
  ENDIF

  s = sTime*64/579*2

  IF (x > XYMax) THEN x = XYMax
  IF (x < XYMin) THEN x = XYMin

  IF (y > XYMax) THEN y = XYMax
  IF (y < XYMin) THEN y = XYMin

  IF (s > SMax) THEN s = SMax

  'Transmit sync pulse (range will be at least halved without this)

  PULSOUT 15, 1200

  'Transmit data with 2-byte start-of-message identifier. The order of the
  'variables is such that the identifier cannot occur twice in message.

  SEROUT Xmt, B2400, [$55, $55, x, s, y]

LOOP
```

The code for the robot reads 14 bytes (all that fits in variable memory) of serial data from the 433MHz transceiver, to increase the chance of capturing a complete 5-byte message. If the 2-byte start-of-message identifier is not found, then there is no data (transmitter is off or out of range), missing data, or the data has been corrupted, and the DAC is set to the last received values. If a certain amount of time elapses when no valid data is received, the DAC will be set to the neutral position (the robot halts). The code is also responsible for managing the startup and shutdown sequences of the robot.

```
'=====================================================================================
'Robot_Rcv.bs2
'Stephen Emery
'05/31/2010
'=====================================================================================

' {$STAMP BS2}
' {$PBASIC 2.5}

'PINS=================================================================================

Rcv       PIN 7   '433 MHz Transceiver

UpDown    PIN 10  'connected to the Up/Down pin of the 3 AD5220s
Clk       PIN 11  'connected to the Clk pin of the 3 AD5220s
CsXPot    PIN 9   'Chip Select for the AD5220 for x joystick output
CsYPot    PIN 8   'Chip Select for the AD5220 for y joystick output
CsSPot    PIN 12  'Chip Select for the AD5220 for throttle output

PwrLatch  PIN 15  'must be set high before the user releases the power button
PbVal     PIN 14  'The state of the robot power button is read on this pin
McOut     PIN 13  '10 ms pulse turns the Motor Controller ON or OFF

LedA      PIN 5   'Bi-color LED anode (used in subroutines)
LedC      PIN 6   'Bi-color LED cathode (used in subroutines)

'CONSTANTS============================================================================

B2400     CON 396 '2400 Baud, 8-Bit, No Parity

InitPos   CON 64  'Initial (center) value of the AD5220s
XYMax     CON 76  'Maximum value for the x and y AD5220s
XYMin     CON 50  'Minimum value for the x and y AD5220s
SMax      CON 127 'Maximum value for the s AD5220

'VARIABLES============================================================================

xPotPos  VAR Byte 'current position of the x AD5220
yPotPos  VAR Byte 'current position of the y AD5220
sPotPos  VAR Byte 'current position of the s AD5220

x  VAR Byte(2)    'present and previous value sent to the x AD5220
y  VAR Byte(2)    'present and previous value sent to the y AD5220
s  VAR Byte(2)    'present and previous value sent to the s AD5220

timeout VAR Word      'used to monitor link status

serData VAR Byte(14)  'data received from the 433MHz Transceiver

i VAR Byte            'counter variable

'MAIN ROUTINE=========================================================================

HIGH PwrLatch         'keeps robot power on
LOW McOut

HIGH CsXPot
HIGH CsYPot
HIGH CsSPot
```

```
xPotPos = InitPos
yPotPos = InitPos
sPotPos = InitPos

FOR i = 0 TO 1
  x(i) = InitPos
  y(i) = InitPos
  s(i) = InitPos
NEXT

timeout = 0

DO
LOOP WHILE (PbVal = 0) 'wait until button is released

PAUSE 200

PULSOUT McOut, 5000    'power up motor controller

DO

  'Get data from 433MHz Transceiver

  SERIN Rcv, B2400, [STR serData\9]

  GOSUB Led_Red

  timeout = timeout + 1

  'Find message

  FOR i = 0 TO 9
    IF ($55 = serData(i) AND $55 = serData(i+1)) THEN
      x(0) = serData(i+2)
      s(0) = serData(i+3)
      y(0) = serData(i+4)
      GOSUB Led_Green        'if no intelligible data, the LED will stay red
      timeout = 0            'clear timeout count
      EXIT
    ENDIF
  NEXT

  'Check data for errors

  IF (XYMin > x(0) OR XYMax < x(0) OR XYMin > y(0) OR XYMax < y(0)) THEN
    x(0) = x(1)
    y(0) = y(1)
  ENDIF

  IF (SMax < s(0)) THEN
    s(0) = s(1)
  ENDIF

  'Check for timeout

  IF (timeout > 10) THEN
    x(0) = InitPos
    y(0) = InitPos
    timeout = 10
  ENDIF

  'Shift values

    x(1) = x(0)
    y(1) = y(0)
    s(1) = s(0)

  GOSUB Check_Pb

  'Set x potentiometer
```

```
  IF (xPotPos > x(0)) THEN
    LOW UpDown
    LOW CsXPot
    DO
      PULSOUT Clk, 1
      xPotPos = xPotPos - 1
    LOOP WHILE (xPotPos <> x(0))
    HIGH CsXPot
  ELSEIF (xPotPos < x(0)) THEN
    HIGH UpDown
    LOW CsXPot
    DO
      PULSOUT Clk, 1
      xPotPos = xPotPos + 1
    LOOP WHILE (xPotPos <> x(0))
    HIGH CsXPot
  ENDIF

  GOSUB Check_Pb

  'Set y potentiometer

  IF (yPotPos > y(0)) THEN
    LOW UpDown
    LOW CsYPot
    DO
      PULSOUT Clk, 1
      yPotPos = yPotPos - 1
    LOOP WHILE (yPotPos <> y(0))
    HIGH CsYPot
  ELSEIF (yPotPos < y(0)) THEN
    HIGH UpDown
    LOW CsYPot
    DO
      PULSOUT Clk, 1
      yPotPos = yPotPos + 1
    LOOP WHILE (yPotPos <> y(0))
    HIGH CsYPot
  ENDIF

  GOSUB Check_Pb

  'Set throttle potentiometer

  IF (sPotPos > s(0)) THEN

    LOW UpDown
    LOW CsSPot
    DO
      PULSOUT Clk, 1
      sPotPos = sPotPos - 1
    LOOP WHILE (sPotPos <> s(0))
    HIGH CsSPot
  ELSEIF (sPotPos < s(0)) THEN
    HIGH UpDown
    LOW CsSPot
    DO
      PULSOUT Clk, 1
      sPotPos = sPotPos + 1
    LOOP WHILE (sPotPos <> s(0))
    HIGH CsSPot
  ENDIF

  GOSUB Check_Pb

LOOP

'SUBROUTINES=========================================================================

Check_Pb:
```

```
  IF (PbVal = 0) THEN
    GOSUB Led_Red
    DO
    LOOP UNTIL (PbVal = 1) 'wait until button is released
    PULSOUT McOut, 5000    'power down motor controller
    PAUSE 200
    LOW PwrLatch           'turn off robot power
  ENDIF

  RETURN

Led_Red:

  HIGH LedC
  LOW LedA

  RETURN

Led_Green:

  LOW LedC
  HIGH LedA

  RETURN
```

Bill of Materials:

Remote:

Description	P/N	MFR/Vendor	Qty.	Ref.
Board of Education (USB)	28850	Parallax	1	-
BASIC Stamp 2	BS2-IC	Parallax	1	U1
433 MHz Transceiver	27982	Parallax	1	U2
2-Axis Joystick	27800	Parallax	1	R5
10K Potentiometer, Trim	152-01031	Parallax	1	R4
Resistor, 220Ω	-	RadioShack	3	R1, R2, R3
Capacitor, 0.1µF	-	-	3	C1, C2, C3

Robot:

Description	P/N	MFR/Vendor	Qty.	Ref.
Shoprider Scooter	-	Shoprider	1	-
5V Regulator	LM7805	RadioShack	1	U1
BASIC Stamp 2	BS2-IC	Parallax	1	U2
433 MHz Transceiver	27982	Parallax	1	U3
10K Digital Potentiometer	AD5220	Parallax	4	U4, U5, U6, U7
Op-Amp	LM741	RadioShack	3	U8, U9, U10
Keyless Entry System	-	Amenity	1	U11
Relay, 12VDC, DPDT	275-249	RadioShack	1	RL1
MOSFET, P-Channel	NTE2381	Fry's Electronics	2	Q1, Q3
Transistor, NPN	2N3904	RadioShack	1	Q2

MOSFET, N-Channel	IRF510	RadioShack	2	Q4, Q5
Diode	1N4005	RadioShack	4	D1, D2, D3, D4
LED, Red	-	RadioShack	1	LED1
LED, Red/Green	-	RadioShack	1	LED2
LED, Yellow	-	RadioShack	1	LED3
LED, Green	-	RadioShack	1	LED4
Pushbutton	-	-	1	SW1
Tact Switch	400-00002	Parallax	1	SW2
Header, 4-pin, Male	-	-	1	J1
Capacitor, 0.33µF	-	-	1	C1
Capacitor, 0.1µF	-	-	1	C2
Resistor, 220Ω	-	RadioShack	1	R6
Resistor, 470Ω	-	RadioShack	1	R5
Resistor, 1K	-	RadioShack	2	R13, R16
Resistor, 10K	-	RadioShack	4	R1, R2, R14, R15
Resistor, 33K	-	RadioShack	3	R7, R8, R9
Resistor, 47K	-	RadioShack	3	R10, R11, R12
Resistor, 100K	-	RadioShack	2	R3, R4
CCFL 12” Blue	801-00010	Parallax	2	Not Shown
12V Inverter for CCFL	750-00060	Parallax	1	Not Shown

Programming Adapter:

Description	**P/N**	**MFR/Vendor**	**Qty.**	**Ref.**
D-Sub, 9-pin, Female	-	Fry’s Electronics	1	P1
Header, 4-pin, Female	-	-	1	P2
Slide Switch, DPDT	275-007	RadioShack	1	SW1
Capacitor, 0.1µF	272-135	RadioShack	2	C1, C2

Pictures:

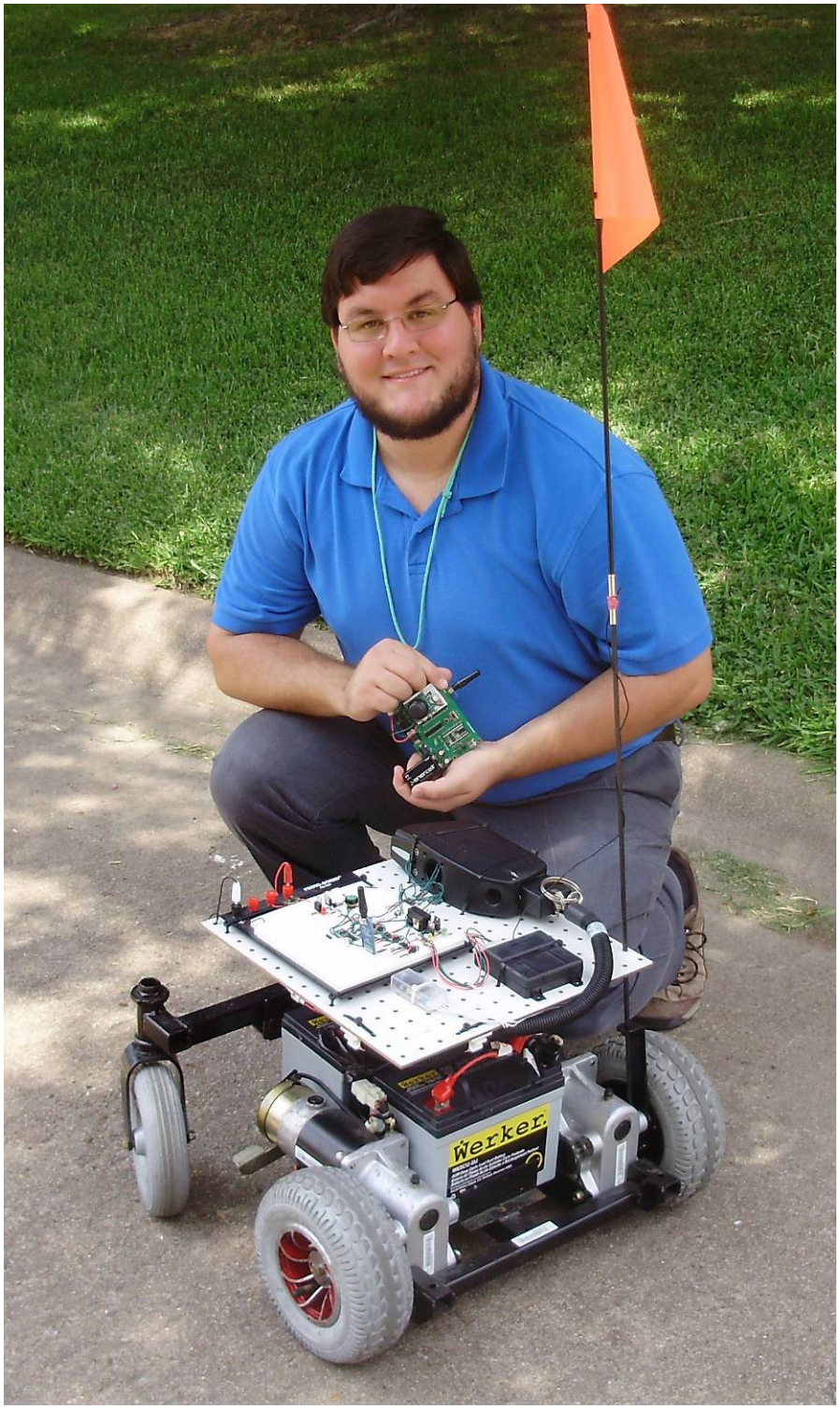

FIGURE 5: Stephen with his Robot

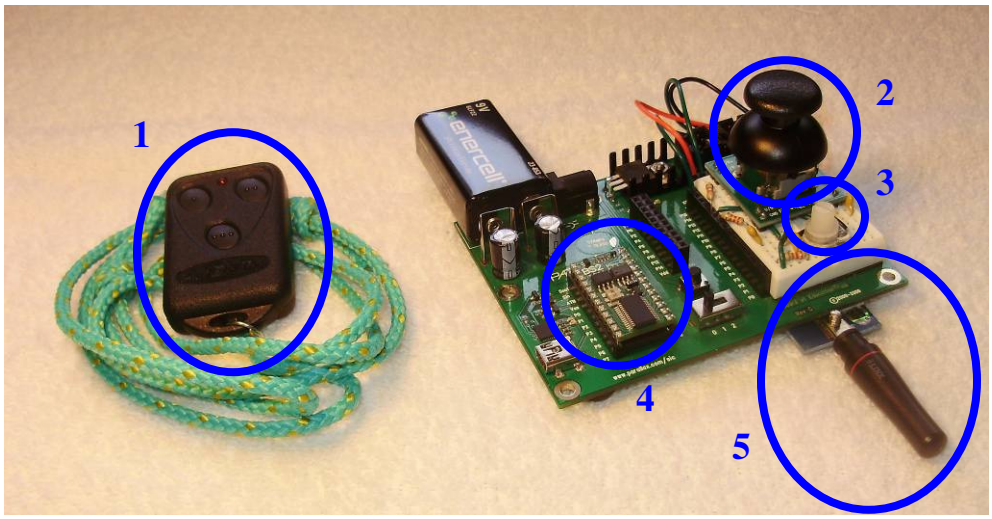

FIGURE 6: E-Stop Fob and Robot Remote Control (Numbers refers to Figure 4)

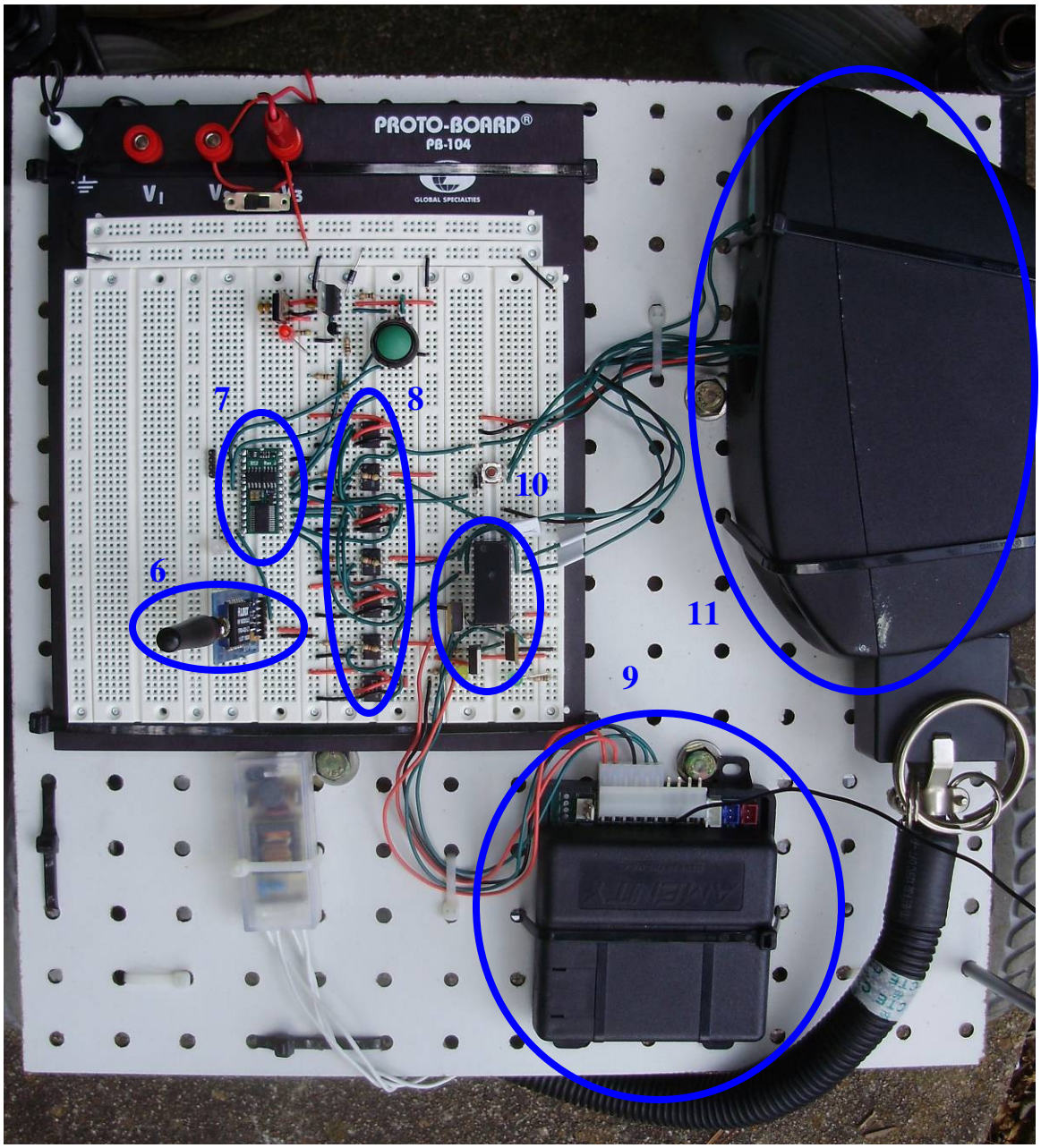

FIGURE 7: Robot Electronics (Numbers refers to Figure 4)

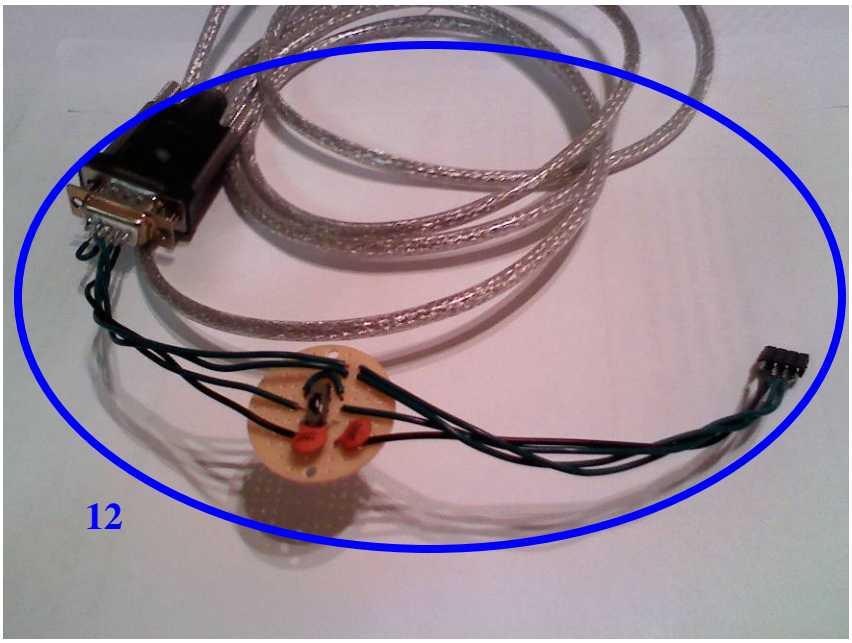

FIGURE 8: Programming Adapter (Numbers refers to Figure 4)

FIGURE 9: The Robot illuminated by its two Cold-Cathode Fluorescent Lamps