Chapter 15. Armatron Robot Control

The Armatron robot arm is one of the coolest toys ever made. Put out by Radio Shack the robot is operated via a couple of levers on the base. Came with all kinds of accessories to pick up and move around. The toy was made by Tomy and later versions showed this name on the box.

Fig. 90 Armatron robot arm

The Armatron we introduce here is micro-programmable version which comes with a separate relay board.

Fig. 91 Micro-programmable version of Armatron robot arm

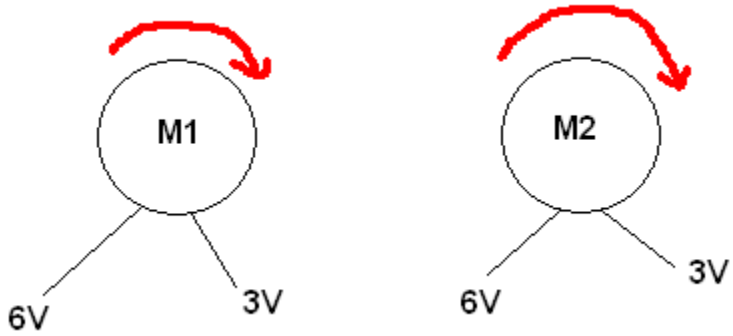
The one we actually have does not have the control pad: it has only the Armatron body and the 7-wire ribbon cable. The focus in the chapter is first to know how to operate the Armatron, and how we develop our own relay board, which in turn can be controlled by the output port of 16F877.

1. Motion Control of the Armatron

The Armatron can move forward, backward, turn right and left, move the arm up and down, and twist, and clamp the jaw. At the bottom there is a big battery housing, and it need 4 D-type 1.5 V dry cell batteries. As soon as you turn on (or by putting the batteries), there are voltages developed at each of the 7 wires in the ribbon cable. This means there is more than slight chance of short circuit unless you separate each wire before turning on the Armatron. Therefore it is safe to separate (on insulate) each wire of the ribbon cable before you inset batteries.

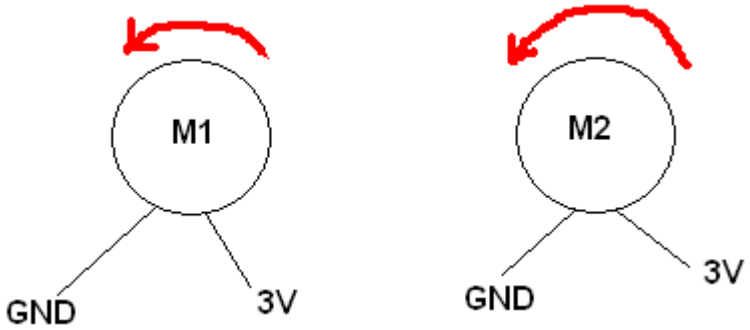
Once the power is on to the Armatron, you have the following voltages developed at each wire:
Black -- 0V (or Ground)
Brown -- 6V
Red --3V
Yellow -- 3V
Orange -- 3V
Green -- 3V
Blue -- 3V

There are motors (DC motors) inside the Armatron, and as you know the direction of the motor turning is controlled by the polarity of the voltage applied to the terminals of the motor. Let's consider an example.



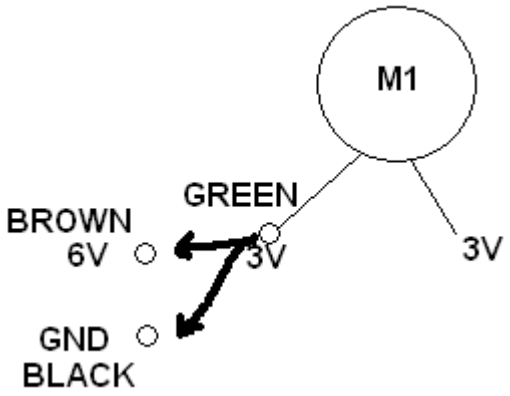
As seen above, if a terminal of the motored is connected to 6V source, and the other terminal is to 3V source, then left terminal higher potential then right, so voltage has positive potential and the motor turns clockwise. If two motors are polarized same, then two would make the body move leftward.

If, instead, the right terminal has higher potential, the motor would turn counter-clockwise, moving the whole body leftward.



This is the principle of controlling the Armatron. Then, how do we change the direction of the motors inside the Armatron? We connect the wires together, based on the original design, we can change the direction of the motor running. Take look at the illustration shown in the next page.

Assume that a motor's left terminal is with 3V. Then the voltage at the other end of also (close to) 3V. Assume again that the left terminal is connected to the green wire of the ribbon cable, while the right terminal is internally connected to the same 3V point. Now as we learned from above, when the power is up, the brown wire develops to 6V, and the black wire is connected to the ground. If the brown wire with 6V potential is connected to the green wire, since brown side is with higher potential, now current flows from the left terminal of the motor to the right.



On the other hand, if the green wire of the motor is connected to the black wire, the right terminal potential with 3V will flow current from right to the left, and this reverses the motor direction.

The overall control pattern is tabulated below:

Armaton Control Diagram
{Basic: connecting two or more wires would turn on motor(s) inside the Armatron. There is a ribbon wire running from the Armatron. The ribbon has 7 colored wires. Controlling the Armatron is to connect the colred wires together. The connecting points are marked by X below}

RIBBON FROM ARMATRON

	Blue	Green	Yellow	Orange	Red	Brown	Black
voltage level*	+3V	+3V	+3V	+3V	+3V	+3V	GND
MOTION							
STOP							
FORWARD		X	X			X	
BACKWARD		X	X				X
TURN RIGHT			X			X	
TURN LEFT			X				X
ARM UP	X					X	
ARM DOWN	X						X
WRIST UP				X		X	
WRIST DOWN				X			X
WRIST TURN					X	X	
CLAMPING					X		X

*Explanation: (1) For FORWARD motion, connect Green, Yellow, and Brown wired together.
(2) For TURN RIGHT motion, connect yellow and Brown wires together.

As explained above, motion control is done by connecting the wires together, and the above table shows exactly how we have to connect which wires together for a desired motion control. If we want to move the Armatron forward, we have to connect green, yellow, and brown wires together. On the other hand, connection of green, yellow, and black wires together would move the Armatron reverse. All the other movements and motions are similarly obtained.

2. Motion Control by Relay

Now let's discuss how we set up a circuit to control the motion by a microcontroller. A relay is an electronic switch. When current flows through the relay coil inside, the magnetic energy generated by the current would grab a metal lever and pull down it. When there is no current and no magnetic energy, my mechanical spring force, the metal lever would stay in the up position touching another contact point. When the metal lever is pulled down, it would touch a contact below.

Let's have an illustration. In the relay shown below left, when there is no voltage source (or Low source) is connected to S, there is no current flowing through the coil, and there is no magnetic force asserted to the metal lever pivoted in the P contact. Then, A and P are opened. This, between A and B, is called "Normal Open" meaning that, at normal condition, A and B are open.

In the relay shown below right, S is the input for relay operation. If S is connected to High side (i.e., +5V for example) then current flows through the coil to the ground. Then, the level pivoted in the P is moved to the left to A. Then, A and P are connected.

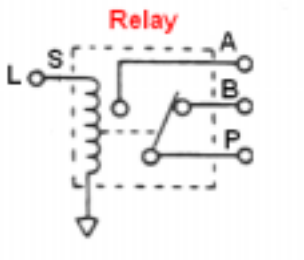


Fig. 92(a) S connected to Low side

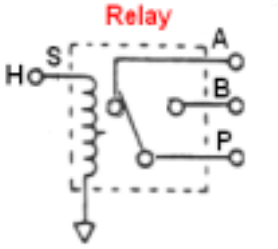


Fig. 92(b) S connected to High side

Now let's expand our discussion on the relay to the motor control case. At the left, we connected

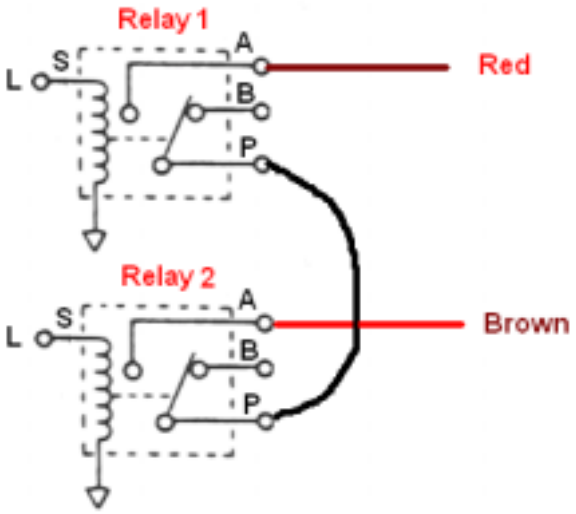


Fig. 93(a) A of the relay 1 and A of the relay 2 disconnected

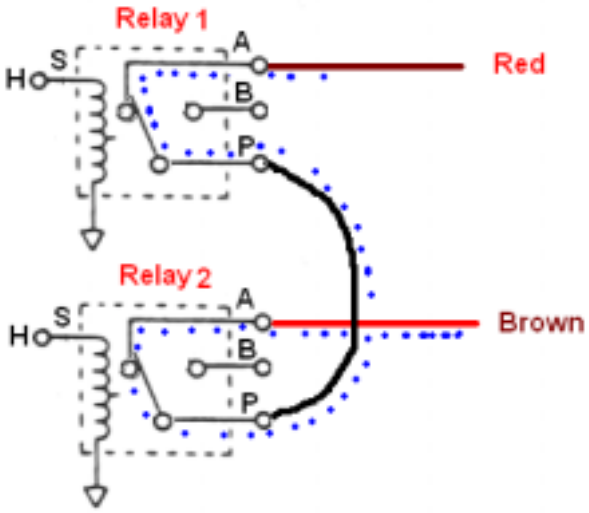


Fig. 93(b) A of the relay 1 and A of relay 2 connected

P points together for both of the relays. At the present situation, with LOW level at the S points, there are no magnetic energy, so A and P are normal open for both relays. This means, A (red) of the relay 1 and the A (brown) of the relay 2 are disconnected.

When we apply High to the S points of the both relay, by means of byte oriented instruction of 16F877, the brown and the red wires are now connected. Expanding this idea to all 7 wires, we can have the following relay configuration.

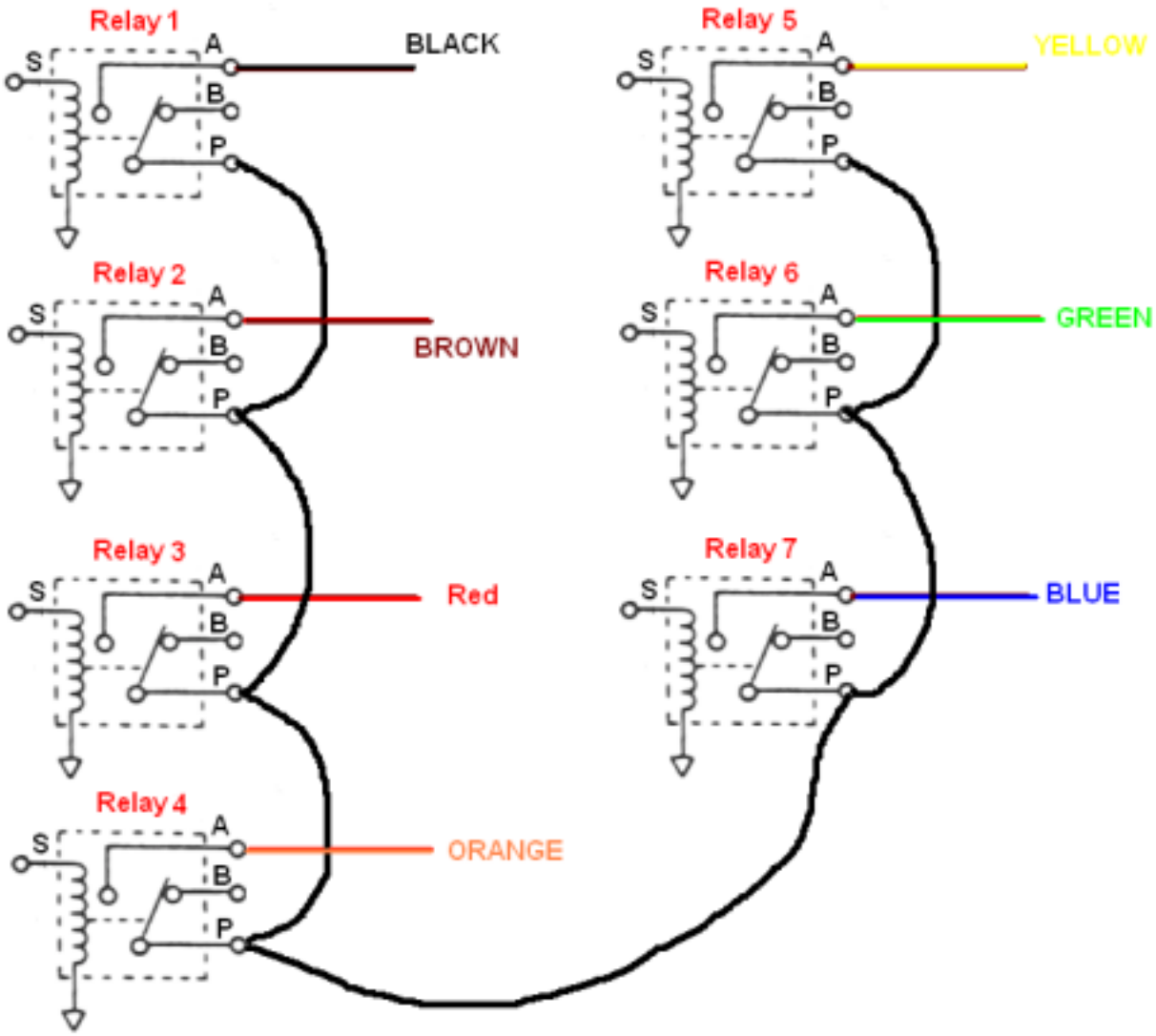


Fig. 94 7 inputs of 7 relays all connected

As we see above, there are 7 inputs, one each to each relay, and this perfectly fits to a port (PORTB, PORTC, or PORTD) of 16F877, and sending out outputs signals 1 or 0 to each line, according to the motion control table, would make the Armatron control easy and simple.

A student implemented this idea into his work, along with other sensors, as depicted next page. In the project, an IR receiver and two IR rangers are installed for IR remote control and collision avoidance.

3. Armatron Control Project

Figure below illustrates the logic flow of the software of the system. The command is first read from the remote control. Safety is the primary concern of the robot motion and this is the first question that must be asked. If it is safe to move then the command can be read and then decoded. If it is unsafe then the command is read but it must be compared to the unsafe direction. If the command coincides with the unsafe direction then it is ignored and the robot awaits a new

command. If the command is safe then it is decoded and then executed. As motion continues safety is continuously checked and also input from the infrared receiver is checked. If any of those two are detected then this loop is broken and the process is restarted. It must be stressed that at every stage of robot motion safety is checked. However, it is only necessary to check safety when the robot is moving forwards and backwards.

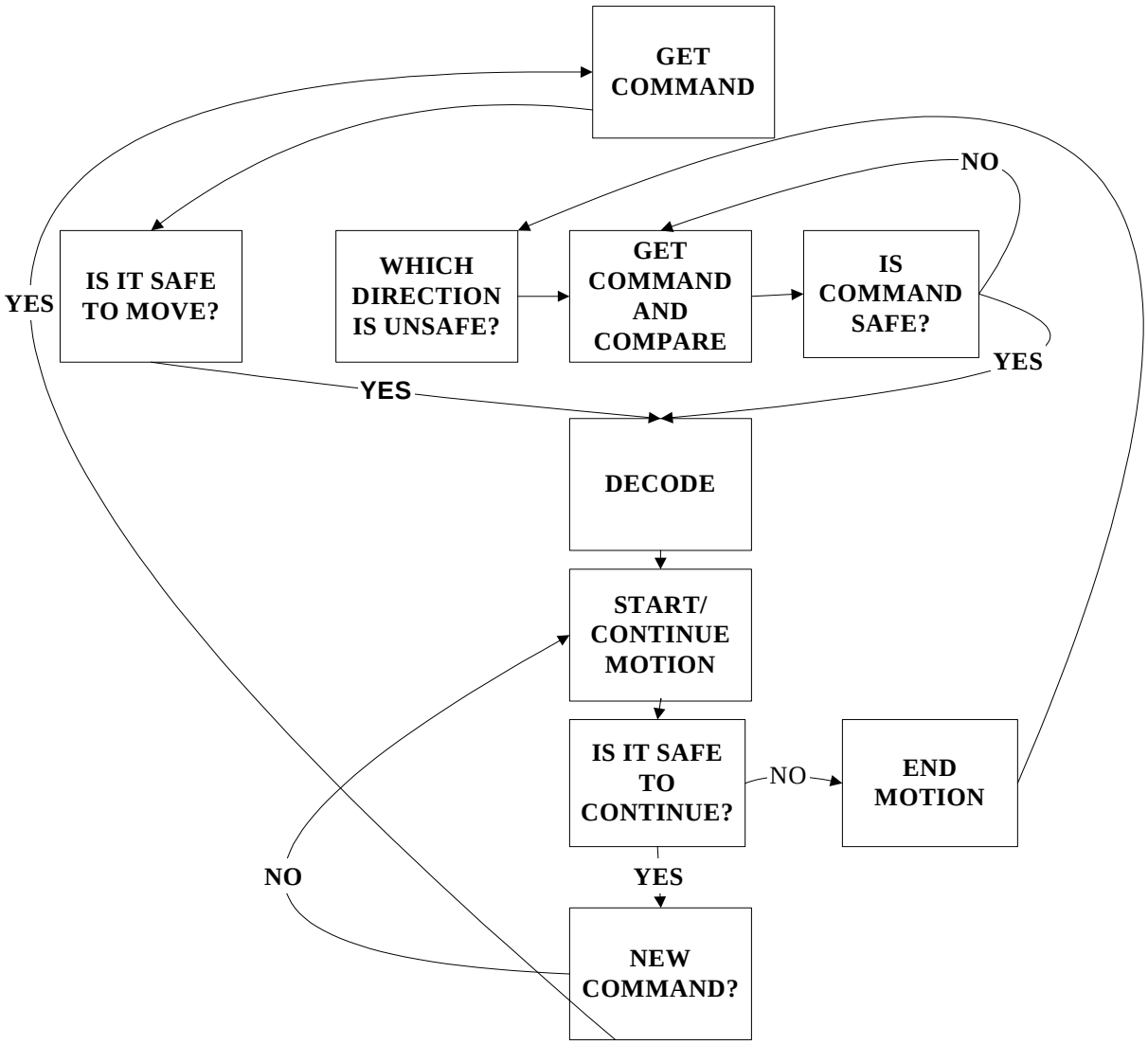


Fig. 95 Software Logic flow of Armatron Robot system

The hardware architecture of the system is shown next. The infrared receiver sends an address and a command to the microprocessor through Port D. The Sony protocol the address has 5 bits and the command 7. These bits are sent in serial form to the microprocessor. These two parts of the signal are stored in two separate registers (see the source code for details). The forward and reverse infrared sensors send their serial outputs to ports RA1 and RA3 respectively. An LED is lighted whenever a signal is received and this is connected to the RA0 output. The input from the infrared receiver determines the output into the relay system. The relay system acts as the interface between the microprocessor and the Armatron robot. Seven outputs are fed from the microprocessor from Port B into the relay system and then into the Armatron robot.

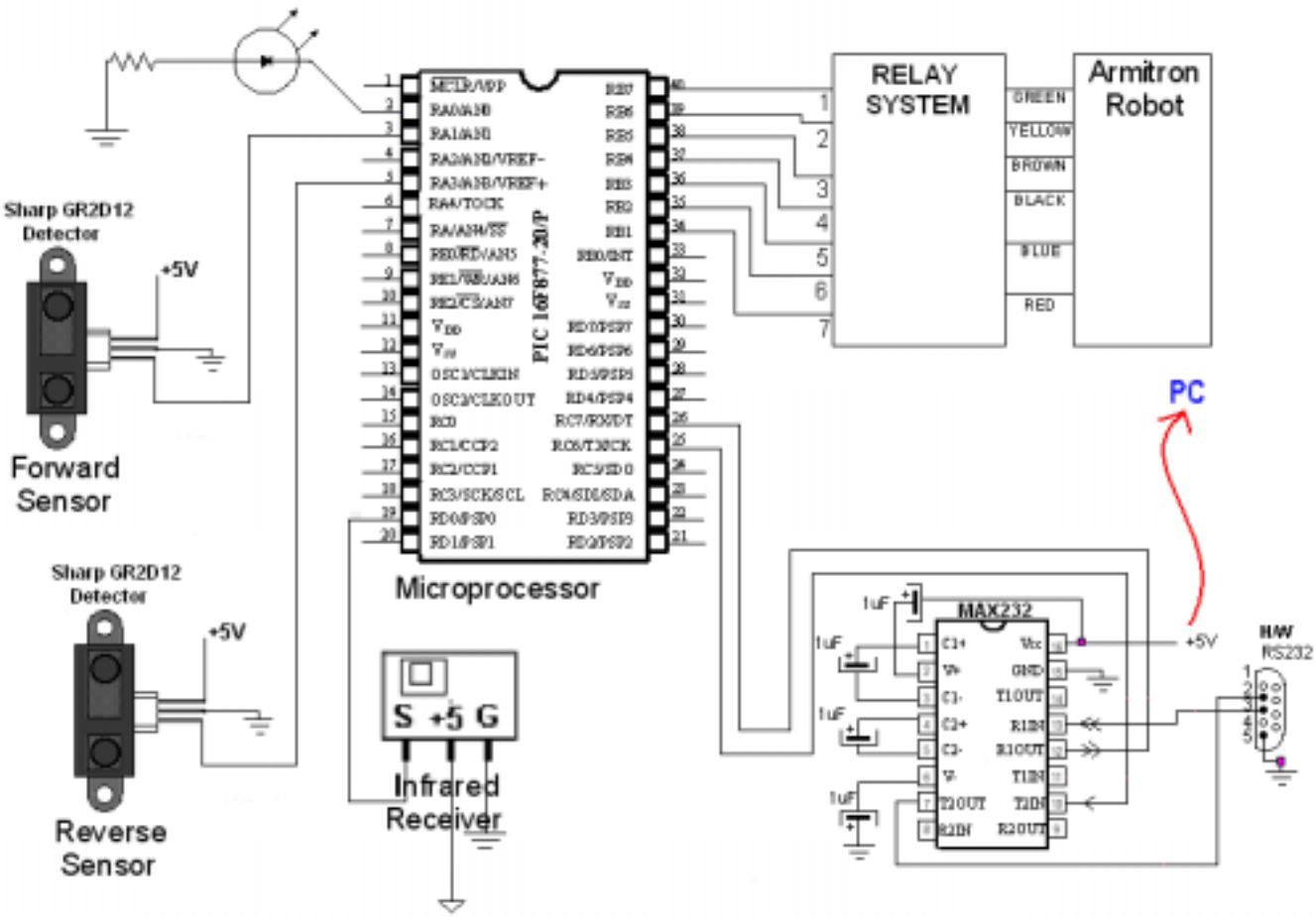


Fig. 96 Hardware architecture of Armatron Robot system

The relay used in the project is depicted next. The relay is a single pole, single throw and is either on or off. Right to it is a schematic of the relay and it shows that when input (left primary side) is received from 16F877, the coil energizes and operates the switch. In this figure wire A and wire B are connected when the coil is energized. The relay is used to close a secondary circuit whenever the primary circuit is energized.

Fig. 97(a) Relay used in the project

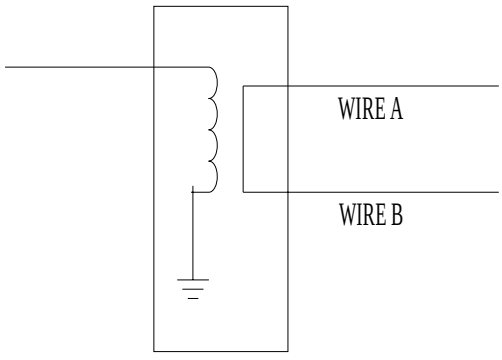


Fig. 97(b) Schematic of the relay

Next is the detail connection diagram for robot control. As discussed before 7 relays are used to execute the commands. The relays are controlled by outputs from Port B of the 16F877 shown coming from the left and the output of the system goes directly to the robot. Those wires are shown on the right.

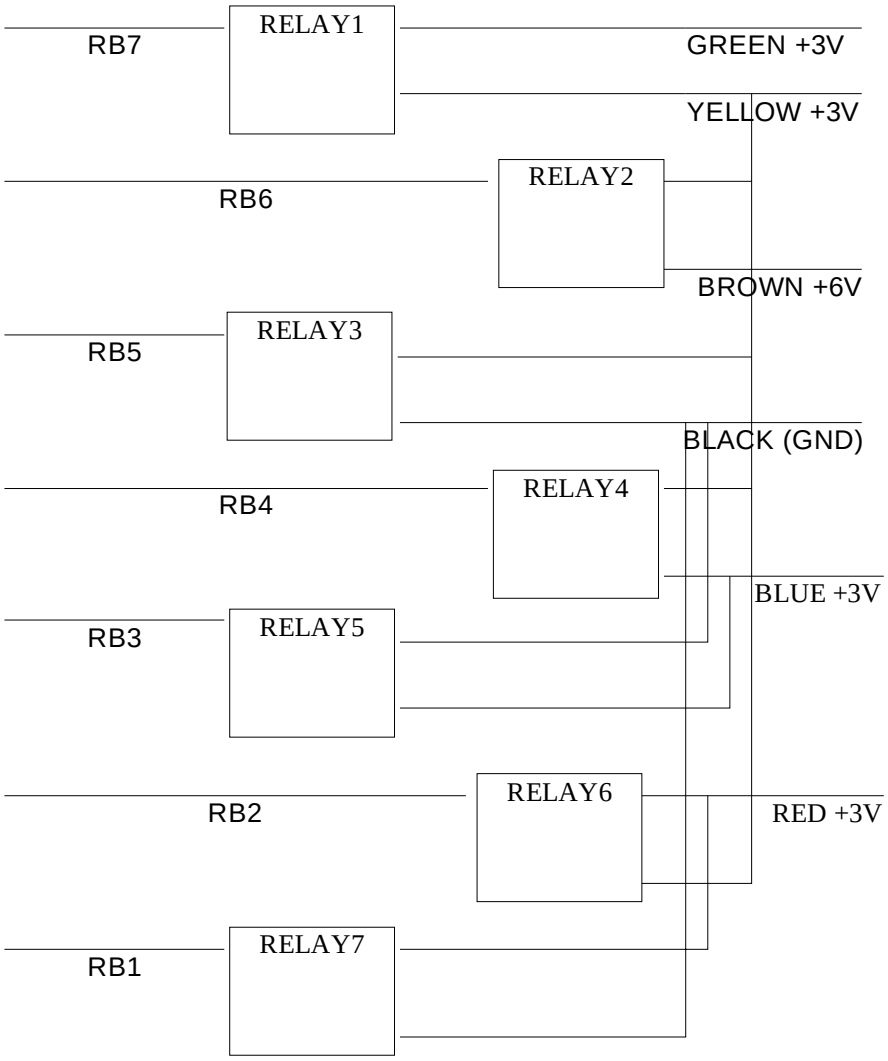


Fig. 98 Detail connection diagram for robot control

The component list for the project is displayed in the table below.

Part Name and Description	**Quantity**	**Part No.**	**Price (each)**	**Total**
PIC 16F877	1	PIC 16F877-20I/P-ND	$9.88	$9.88
RS232 Transceiver 16-DIP	1	MAX232CPE-ND	$3.31	$3.31
1Uf 50V Electrolytic Capacitor	4	P1196-ND	$0.37	$1.48
20 MHz Crystal Oscillator	1	CTX062-ND	$0.94	$0.94
20pF(22pF) Ceramic Oscillator	2	1330PH-ND	$0.72	$1.46

DB-9 Connector, Female	1	A2100-ND	$2.36	$2.36
9-pin Serial Cable, Male-Female	1	AE1020-ND (2m long)	$5.35	$5.35
40-Pin Wire Wrap Socket	1	ED4640-ND	$4.38	$4.38
16-Pin Wire Wrap Socket	1	ED4316-ND	$1.38	$1.38
Prototype Board	1	V2012-ND(6''X6'')	$9.36	$9.36
7805 Voltage Regulator, TO-220	1	NJM7805FA-ND	$0.60	$0.60
SPST Reed Relays	7	275-232	$2.14	$14.98
Infrared Sensors	2	GP2D12	$11.59	$23.18
Project Box	1		$5.39	$5.39
Infrared Remote Control	1	15-2131	$14.59	$14.59
Infrared Receiver	1		$2.00	$2.00
TOTAL				**$101.28**

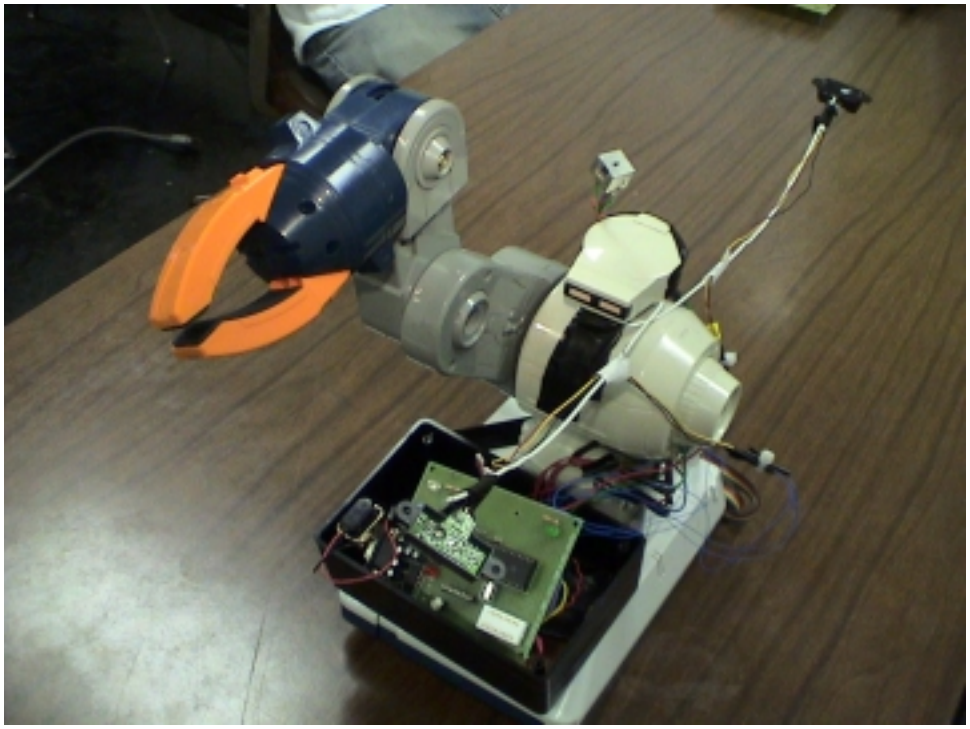


Fig. 99 The completed Armatron Robot

4. Source Code

**Note: This source code is written by Jason Burrows who took the Embedded Computing course from me. I revised/added slightly only the parts I think are better for beginners to understand easily. You can find the routines we discussed in the previous chapters here in the source code.*

```
;JASON BURROWES-JONES
;ARMITRON ROBOT ARM CONTROL


        LIST P = 16F877

;*****************************************************************************
;REGISTER DECLARATIONS
;*****************************************************************************

STATUS      EQU   0X03
PORTA EQU   0X05
PORTB EQU   0X06
PORTD EQU   0X08
TRISA EQU   0X85
TRISB EQU   0X86
TRISD EQU   0X88
PIR1  EQU   0X0C        ;PERIPHERAL INTERRUPT FLAG
                        ;LOCATED AT THIS ADDRESS AS
                        ;SPECIFIED IN MANUAL
RCIF  EQU   0X05        ;RECEIVE INTERRUPT FLAG
                        ;1 RECEIVE BUFFER IS FULL
                        ;0 REVEIVE BUFFER IN EMPTY
TXIF  EQU   0X04        ;TRANSMIT INTERRUPT FLAG
                        ;1 TRANSMIT BUFFER IS EMPTY
                        ;0 TRANSMIT BUFFER IS FULL
BAUD  EQU   0X0F        ;BAUD RATE IS 15 FOR 19200 BPS
PIE1  EQU   0X8C
PIR1  EQU   0X0C
ADCON0      EQU   0X1F
ADCON1      EQU   0X9F
ADRESH      EQU   0X1E
ADRESL      EQU   0X9E

ADIE  EQU   0X06
ADIF  EQU   0X06
GO          EQU   0X02

IRX         EQU   0X00        ;INFRARED INPUT BIT IN PORTD
CARRY EQU   0X00        ;CARRY BIT OF STATUS REGISTER
MSB         EQU   0X07        ;MOST SIG BIT OF REGISTER
ZFLAG EQU   0X02        ;ZERO FLAG
BUZ         EQU   0X04        ;BUZZER CONNECTED TO PORT<D4>
BIT3  EQU   0X03
BIT2  EQU   0X02
BIT1  EQU   0X01

;*****************************************************************************
;DATA SPACE FOR RAM
;*****************************************************************************
```

```
        CBLOCK    0X20              ;RAM AREA FOR USE AT ADDRESS 20H
                        TEMP        ;STORES TEMP VARIABLE
                        FIRST       ;VARIABLE DECLARATION
                        SECOND
                        THIRD
                        COUNT
                        DCOUNT
                        CMCOUNT           ;KEEPS COUNT OF NUMBER OF BITS IN
COMMAND
                        ADDCOUNT    ;KEEPS COUNT OF NUMBER OF BITS IN ADDRESS
                        PCOUNT            ;KEEPS COUNT IF NUMBER OF
120MICROSECS
                        LCOUNT            ;KEEPS COUNT OF NUMBER OF LED
FLASHES
                        COMREG            ;REGISTER STORES COMMAND BITS
                        ADDREG            ;REGISTER STORES ADDRESS BITS
                        SAFETY_FLAG
                        FR_FLAG
                        N120US
                        DIGIT1
                        DIGIT2
                        DIGIT3
            ENDC                    ;END OF RAM BLOCK
;#################################################################################

      ORG         0X00
      GOTO        MAIN
      ORG         0X06

;*********************************************************************************
;#################################################################################
;MAIN PROGRAM
;#################################################################################
;*********************************************************************************

MAIN
      MOVLW       0X55
      MOVWF       DCOUNT
      CALL        DELAY                       ;DELAY TO WARM UP HARDWARE
      CALL        INIT
      CALL        STOP                    ;MAKE SURE THE ROBOT IS NOT MOVING

INITIAL_READ_INFRARED
      CALL  INFRARED_READ               ;READ INPUT FROM REMOTE CONTROL
      CALL  INPUT_TEST                  ;TEST FOR BAD INPUT
      BTFSC TEMP,BIT1         ;IF THE INPUT IS BAD THEN CONTINUE TO READ
                              ;AND WAIT FOR GOOD INPUT
      GOTO  INITIAL_READ_INFRARED

FR_DETERMINE
      CALL  FR_FLAG_SET       ;THIS DETERMINES WHETHER A FORWARD OR REVERSE
                              ;HAS BEEN GIVEN

      CALL  MOTION_SAFETY     ;AFTER RECEIVING THE COMMAND THE SENSORS ARE
                              ;CHECKED TO DETERMINE ANY DANGER

PERFORM_OPERATION
```

```
      CALL  DET_PERFORM_OPERATION   ;DETERMINE THE OPERATION TO BE PERFORMED
                                    ;AND PERFORM THIS OPERATION IF THERE IS NO
                                    ;COMFLICT WITH THE SENSORS


;THE FOLLOWING CONTINUOUSLY CHECKS FOR NEW
;INPUT AND TO SEE IF THE ROBOT SENSES ANY DANGER

CHECKING
      CALL  MOTION_SAFETY                ;CHECK SAFETY
      BANKSEL     PORTD
      BTFSC PORTD, IRX                   ;TEST FOR IR INPUT AND START BIT
      GOTO  CHECKING

CONTINUOUS_INFRARED_READ
      CALL  INFRARED_READ                ;READ INPUT FROM REMOTE CONTROL
      CALL  INPUT_TEST                   ;TEST FOR BAD INPUT
      BTFSC TEMP,BIT1
      GOTO  CHECKING
      GOTO  FR_DETERMINE


;*****************************************************************************
;#############################################################################
;END OF MAIN PROGRAM
;#############################################################################
;*****************************************************************************


;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
;*****************************************************************************
;THIS SUBROUTINE READS THE INFRARED SIGNAL FROM REMOTE CONTROLLER
;*****************************************************************************
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

INFRARED_READ

      BANKSEL     CMCOUNT
      MOVLW 0X07              ;COMMAND HAS SEVEN BITS
      MOVWF CMCOUNT
WAIT
      BTFSS PORTD, IRX        ;WAIT FOR SEPERATOR
                                    ;SHOULD BE 600 MICROSECS
      GOTO  WAIT
CMNEXT
      CLRF  PCOUNT                  ;NUMBER OF 120 MICROSEC DURATIONS
      BCF         STATUS, CARRY     ;CLEAR CARRY BIT OF STATUS
      RRF         COMREG                  ;COMMAND STORAGE, MSB IS NOW 0
WAIT2
      BTFSC PORTD, IRX        ;TESTS AND WAITS FOR THE END OF
                              ;SEPERATOR
      GOTO  WAIT2
DURATION
      CALL  DELAY120US        ;120 MICROSECS DELAY SUBROUTINE
WAIT3
      BTFSC PORTD, IRX
      GOTO  ONEZERO           ;DETERMINES END OF LOW DURATION
```

```
                                    ;IS IT A 1 OR 0
      INCF  PCOUNT                  ;DETERMINES NUMBER OF 120US
      GOTO  DURATION
ONEZERO
      BTFSC PCOUNT,0X03       ;IF GREATER THAN 8 THEN WE HAVE A
                                    ;120US DELAY AND WE LEAVE MSB AS IS
      BSF         COMREG,MSB        ;OTHERWISE WE SET MSB AS 1
      DECFSZ      CMCOUNT
      GOTO  CMNEXT
      BCF         STATUS,CARRY
      RRF         COMREG
ADD_READ
      CLRF  ADDREG
      BANKSEL     ADDCOUNT
      MOVLW 0X05              ;COMMAND HAS FIVE BITS
      MOVWF ADDCOUNT
ADDNEXT
      CLRF  PCOUNT                  ;NUMBER OF 120 MICROSEC DURATIONS
      BCF         STATUS, CARRY     ;CLEAR CARRY BIT OF STATUS
      RRF         ADDREG            ;COMMAND STORAGE, MSB IS NOW 0
WAITB
      BTFSC PORTD, IRX        ;TESTS AND WAITS FOR THE END OF
                              ;SEPERATOR
      GOTO  WAITB
DURATIONA
      CALL  DELAY120US        ;120 MICROSECS DELAY SUBROUTINE
WAITC
      BTFSC PORTD, IRX
      GOTO  ONEZERO_A         ;DETERMINES END OF LOW DURATION
                                    ;IS IT A 1 OR 0
      INCF  PCOUNT                  ;DETERMINES NUMBER OF 120US
      GOTO  DURATIONA
ONEZERO_A
      BTFSC PCOUNT,0X03       ;IF GREATER THAN 8 THEN WE HAVE A
                                    ;120US DELAY AND WE LEAVE MSB AS IS
      BSF         ADDREG,MSB        ;OTHERWISE WE SET MSB AS 1
      DECFSZ      ADDCOUNT
      GOTO  ADDNEXT
      BCF         STATUS,CARRY      ;SET CARRY BIT TO ZERO
      BANKSEL     ADDREG
      RRF         ADDREG
      RRF         ADDREG
      RRF         ADDREG
      RETURN

;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
;*****************************************************************
;THIS SUBROUTINE CHECKS SAFETY DURING MOTION
;*****************************************************************
MOTION_SAFETY
      CALL  CHECK_SENSORS           ;CHECK SENSOR FOR SAFETY
      BTFSS SAFETY_FLAG,BIT1  ;IF SAFE SIMPLY RETURN
      RETURN

      BTFSS SAFETY_FLAG,BIT2  ;IF BIT 2 IS SET THEN STOP
                                    ;FORWARD MOTION
```

```
      GOTO  REVERSE_STOP            ;OTHERWISE CHECK REVERSE MOTION

FORWARD_STOP
      BANKSEL     FR_FLAG           ;TEST FR FLAG AND IF BIT 1 IS SET THEN
                                    ;IT THE ROBOT IS MOVING FORWARD
      BTFSC FR_FLAG,BIT1
      CALL  STOP
      RETURN

REVERSE_STOP
      BTFSS SAFETY_FLAG,BIT3  ;IF BIT 3 IS SET THEN IT IS UNSAFE TO
                              ;MOVE BACKWARDS AND THEN WE MOVE BACKWARDS
      RETURN
      BANKSEL     FR_FLAG     ;CHECK FR FLAG TO SEE IF THE ROBOT IS
                              ;MOVING FORWARDS OR BACKWARDS
      BTFSC FR_FLAG,BIT2
      CALL  STOP
      RETURN

;*********************************************************************
;THIS SUBROUTINE READS THE SENSORS AND DETERMINES SAFE AND UNSAFE MOTION
;*********************************************************************

CHECK_SENSORS
      CLRF  SAFETY_FLAG             ;ALWAYS ASSUME SAFE UNTIL PROVEN
                                          ;OTHERWISE
SENSOR1
      CALL  SENSOR1_INIT
      CALL  SENSOR_READ             ;READ FORWARD SENSOR
      CALL  SAFETY_CHECK            ;CHECK FORWARD SAFETY
      BTFSS SAFETY_FLAG,BIT1  ;IF FORWARD SENSOR IS UNSAFE THEN SET
                                    ;BIT 2 OF THE SAFETY BIT AND THEN
                                          ;RETURN
      GOTO  SENSOR2                 ;IF NOT THEN CHECK THE SECOND SENSOR

FORWARD_UNSAFE                            ;IF FORWARD UNSAFE THEN SET BIT2
      BANKSEL     SAFETY_FLAG
      BSF         SAFETY_FLAG,BIT2
      RETURN

SENSOR2
      CALL  SENSOR2_INIT
      CALL  SENSOR_READ             ;READ REVERSE SENSOR
      CALL  SAFETY_CHECK            ;CHECK REVERSE SAFETY
      BTFSS SAFETY_FLAG,BIT1  ;IF REVERSE SENSOR IS UNSAFE THEN SET
                                    ;BIT 3 OF THE SAFETY BIT AND THEN
                                          ;RETURN
      RETURN                              ;IF NOT THEN SIMPLY RETURN

REVERSE_UNSAFE                            ;IF REVERSE UNSAFE THEN SET BIT3
      BANKSEL     SAFETY_FLAG
      BSF         SAFETY_FLAG,BIT3
      RETURN

;*********************************************************************
;SUBROUTINE INITIALIZES SENSOR1
;*********************************************************************
```

```
SENSOR1_INIT
      BANKSEL     PIE1
      BCF         PIE1,ADIE   ;DISABLE ADC INTERRUPT

      BANKSEL     PIR1
      BCF         PIR1,ADIF

      BANKSEL     ADCON0
      MOVLW 0XD9
      MOVWF ADCON0            ;INITIALISE ADC(RA1 IS ADC PORT)

      BANKSEL     ADCON1
      MOVLW 0X00
      MOVWF ADCON1            ;PORTA IS ADC CHANNEL LEFT JUSTIFIED
                              ;BITS 8 AND 9 ARE IN ADRESH
                              ;BITS 7 TO 0 ARE IN ADRESL

      RETURN

;*************************************************************************
;SUBROUTINE INITIALISES SENSOR2
;*************************************************************************
SENSOR2_INIT
      BANKSEL     PIE1
      BCF         PIE1,ADIE   ;DISABLE ADC INTERRUPT

      BANKSEL     PIR1
      BCF         PIR1,ADIF

      BANKSEL     ADCON0
      MOVLW 0XC9        ;CHANGE TO SENSOR 2 INPUT PORT
      MOVWF ADCON0            ;INITIALISE ADC(RA2 IS ADC PORT)

      BANKSEL     ADCON1
      MOVLW 0X00
      MOVWF ADCON1            ;PORTA IS ADC CHANNEL LEFT JUSTIFIED
                                    ;BITS 8 AND 9 ARE IN ADRESH
                                    ;BITS 7 TO 0 ARE IN ADRESL
      MOVLW 0X25
      MOVWF DCOUNT
      CALL  DELAY
      RETURN

;*********************************************************************
;THIS SUBROUTINE READS THE SENSOR
;*********************************************************************

SENSOR_READ
      BANKSEL     ADCON0
      BSF         ADCON0, GO        ;START CONVERSION

ADCLOOP
      BANKSEL     ADCON0
      BTFSC ADCON0, GO        ;WAIT UNTIL DATA FULLY COLLECTED
      GOTO  ADCLOOP

      BANKSEL     PIR1
```

```
      BCF         PIR1,ADIF         ;CLEAR CONVERSION COMPLETE FLAG

      BANKSEL     ADRESH
      MOVF  ADRESH,W

      MOVWF TEMP

      RETURN

;*********************************************************************
;THIS SUBROUTINE CHECKS MOTION SAFETY
;*********************************************************************

SAFETY_CHECK
      BTFSC TEMP,0X07         ;10CM
      GOTO  SAFE
      BTFSC TEMP,0X06         ;20CM
      GOTO  SAFE
      BTFSC TEMP,0X05         ;40CM
      GOTO  SAFE
      GOTO  UNSAFE
SAFE
      CLRF  SAFETY_FLAG
      CLRF  TEMP
      RETURN
UNSAFE
      BSF         SAFETY_FLAG,BIT1
      CLRF  TEMP
      RETURN

;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
;*********************************************************************

;*********************************************************************
;THIS SUBROUTINE DETERMINES WHICH OPERATION SHOULD TAKE PLACE
;*********************************************************************

DET_PERFORM_OPERATION

      BANKSEL     ADDREG                  ;TEST FOR VIDEO COMMAND
      MOVF  ADDREG,W
      XORLW H'07'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      GOTO  VIDEO

      BANKSEL     ADDREG                  ;TEST FOR TV COMMAND
      MOVF  ADDREG,W
      XORLW H'01'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      GOTO  TV
      GOTO  BAD_INPUT

TV
LEFT
      BANKSEL     COMREG                  ;SHOULD WE TURN LEFT?
```

```
      MOVF  COMREG,W
      XORLW H'13'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  TURN_LEFT
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

RIGHT
      BANKSEL     COMREG                ;SHOULD WE TURN RIGHT?
      MOVF  COMREG,W
      XORLW H'12'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  TURN_RIGHT
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

VIDEO
      BANKSEL     COMREG                ;SHOULD WE GO FORWARD?
      MOVF  COMREG,W
      XORLW H'1C'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  FORWARD
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN


      BANKSEL     COMREG                ;SHOULD WE REVERSE?
      MOVF  COMREG,W
      XORLW H'1B'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  REVERSE
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN


      BANKSEL     COMREG                ;SHOULD WE STOP?
      MOVF  COMREG,W
      XORLW H'18'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  STOP
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN


      BANKSEL     COMREG                ;SHOULD WE MOVE ARM UP?
      MOVF  COMREG,W
      XORLW H'10'
```

```
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  ARM_UP
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

      BANKSEL     COMREG                  ;SHOULD WE MOVE ARM DOWN?
      MOVF  COMREG,W
      XORLW H'11'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  ARM_DOWN
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

      BANKSEL     COMREG                  ;SHOULD WE TURN WRIST?
      MOVF  COMREG,W
      XORLW H'3F'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  WRIST_TURN
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

      BANKSEL     COMREG                  ;SHOULD WE CLAMP?
      MOVF  COMREG,W
      XORLW H'2A'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  CLAMP
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

      BANKSEL     COMREG                  ;SHOULD WE EXECUTE A ROUTINE?
      MOVF  COMREG,W
      XORLW H'1A'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  PRESET_ROUTINE_DET
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN


BAD
      CALL  BAD_INPUT   ;IF NONE OF THE CONDITIONS ARE
                                ;SATISFIED THEN WE BUZZ
      RETURN
```

```
;*********************************************************************
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
;THE FOLLOWING SUBROUTINES CAUSES THE SPECIFIED MOTION IN THE ROBOT
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
;*********************************************************************

;*********************************************************************
TURN_LEFT
;*********************************************************************
      BANKSEL     PORTB
      MOVLW H'08'
      MOVWF PORTB
      RETURN
;*********************************************************************
TURN_RIGHT
;*********************************************************************
      BANKSEL     PORTB
      MOVLW H'04'
      MOVWF PORTB
      RETURN
;*********************************************************************
FORWARD
;*********************************************************************
      BANKSEL     SAFETY_FLAG             ;IS FORWARD MOTION SAFE?
      BTFSS SAFETY_FLAG,BIT1
      GOTO  FORWARD_MOTION
      BTFSS SAFETY_FLAG,BIT2
      GOTO  FORWARD_MOTION
      RETURN
FORWARD_MOTION
      BANKSEL     PORTB
      MOVLW H'06'
      MOVWF PORTB
      BANKSEL     FR_FLAG
      BSF         FR_FLAG,BIT1
      RETURN
;*********************************************************************
REVERSE
;*********************************************************************
      BANKSEL     SAFETY_FLAG             ;IS REVERSE MOTION SAFE?
      BTFSS SAFETY_FLAG,BIT1
      GOTO  REVERSE_MOTION
      BTFSS SAFETY_FLAG,BIT3
      GOTO  REVERSE_MOTION
      RETURN
REVERSE_MOTION
      BANKSEL     PORTB
      MOVLW H'0A'
      MOVWF PORTB
      BANKSEL     FR_FLAG
      BSF         FR_FLAG,BIT2
      RETURN
```

```
;*********************************************************************
STOP
;*********************************************************************
      BANKSEL     PORTA
      MOVLW H'00'
      MOVWF PORTA
      BANKSEL     PORTB
      MOVLW H'00'
      MOVWF PORTB
      RETURN
;*********************************************************************
ARM_UP
;*********************************************************************
      BANKSEL     PORTB
      MOVLW H'10'
      MOVWF PORTB
      RETURN
;*********************************************************************
ARM_DOWN
;*********************************************************************
      BANKSEL     PORTB
      MOVLW H'20'
      MOVWF PORTB
      RETURN
;*********************************************************************
WRIST_TURN
;*********************************************************************
      BANKSEL     PORTB
      MOVLW H'40'
      MOVWF PORTB
      RETURN
;*********************************************************************
CLAMP
;*********************************************************************
      BANKSEL     PORTA
      MOVLW H'20'
      MOVWF PORTA
      RETURN

;*********************************************************************
PRESET_ROUTINE_DET
;*********************************************************************
                                          ;ANY MOTION IS FIRST STOPPED
                                          ;BEFORE PRESET SUBROUTINE IS CHOSEN
                                          ;THERE MUST BE TWO BUZZES
      CALL  STOP
      CALL  DELAY
      CALL  BUZZ
      CALL  DELAY
      CALL  BUZZ

PLAY_WHAT
      CALL  INFRARED_READ
      BANKSEL     ADDREG                      ;TEST FOR VIDEO COMMAND
      MOVF  ADDREG,W
      XORLW H'07'
      BANKSEL     STATUS
```

```
      BTFSS STATUS,ZFLAG
      GOTO  PLAY_WHAT

      BANKSEL     COMREG                        ;SHOULD WE PLAY ROUTINE 1?
      MOVF  COMREG,W
      XORLW H'00'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  ROUTINE1
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

      BANKSEL     COMREG                        ;SHOULD WE PLAY ROUTINE 2?
      MOVF  COMREG,W
      XORLW H'01'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  ROUTINE2
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN


      BANKSEL     COMREG                        ;SHOULD WE PLAY ROUTINE 3?
      MOVF  COMREG,W
      XORLW H'02'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  ROUTINE2
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

      BANKSEL     COMREG                        ;SHOULD WE PLAY ROUTINE 4?
      MOVF  COMREG,W
      XORLW H'03'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  ROUTINE2
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

      BANKSEL     COMREG                        ;SHOULD WE PLAY ROUTINE 5?
      MOVF  COMREG,W
      XORLW H'04'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      CALL  ROUTINE2
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      RETURN

      RETURN
;*****************************************************************
ROUTINE1
```

```
;THIS ROUTINE JUST SENDS THE ROBOT FORWARD AND THEN BACKWARDS AND THEN
;STOPS
;*********************************************************************
      MOVLW 0XA0
      MOVWF DCOUNT
      CALL  FORWARD
      CALL  MOTION_DELAY
      CALL  STOP
      CALL  REVERSE
      CALL  MOTION_DELAY
      CALL  STOP
      RETURN


;*********************************************************************
ROUTINE2
;*********************************************************************

      RETURN

;*********************************************************************
ROUTINE3
;*********************************************************************

      RETURN

;*********************************************************************
ROUTINE4
;*********************************************************************

      RETURN

;*********************************************************************
ROUTINE5
;*********************************************************************

      RETURN




;*********************************************************************
BAD_INPUT
      ;CALL BUZZ
      RETURN

;*********************************************************************
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
;*********************************************************************
;*************************************************************************
******
;SUBROUTINE INITIALIZES REGISTERS
;*************************************************************************
******

INIT
      BANKSEL     TRISA
```

```
      MOVLW 0X0F
      MOVWF TRISA

      BANKSEL     TRISB
      MOVLW 0X00
      MOVWF TRISB

      BANKSEL     TRISD
      MOVLW 0X01
      MOVWF TRISD

;*********************************************************************
;DELAY 120 MICROSECS SUBROUTINE
;*********************************************************************

DELAY120US
      MOVLW 0XC5
      MOVWF N120US
AGAIN
      DECFSZ      N120US
      GOTO  AGAIN
      RETURN

;*********************************************************************
;DELAY SUBROUTINE
;*********************************************************************

DELAY
      MOVF  DCOUNT,W
    MOVWF   FIRST
DLOOP
      MOVWF SECOND
      DECFSZ      FIRST
      GOTO  NEXT1
      GOTO  THEEND
NEXT1
      MOVWF THIRD
      DECFSZ      SECOND
      GOTO  NEXT2
      GOTO  DLOOP
NEXT2
      DECFSZ      THIRD
      GOTO  NEXT2
      GOTO  NEXT1
THEEND
      RETURN

;*********************************************************************
;INPUT TEST SUBROUTINE
;*********************************************************************

INPUT_TEST

      MOVF  ADDREG,W
      XORLW H'07'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
```

```
      GOTO  GOODINPUT

      MOVF  ADDREG,W
      XORLW H'01'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      GOTO  GOODINPUT
      GOTO  BADINPUT

GOODINPUT
      CLRF  TEMP
      RETURN
BADINPUT
      BSF         TEMP,BIT1
      RETURN


;*********************************************************************
;THIS SUBROUTINE DETERMINES WHETHER FORWARD OF REVERSE COMMAND IS GIVEN
;*********************************************************************
FR_FLAG_SET

      BANKSEL     FR_FLAG
      CLRF  FR_FLAG

      BANKSEL     COMREG
      MOVF  COMREG,W
      XORLW H'1C'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      GOTO  FORWARD_SET

      BANKSEL     COMREG
      MOVF  COMREG,W
      XORLW H'1B'
      BANKSEL     STATUS
      BTFSC STATUS,ZFLAG
      GOTO  REVERSE_SET
      BANKSEL     FR_FLAG
      CLRF  FR_FLAG
      RETURN

FORWARD_SET
      BSF         FR_FLAG,BIT1
      RETURN
REVERSE_SET
      BSF         FR_FLAG,BIT2
      RETURN


;*********************************************************************
;MOTION DELAY SUBROUTINE
;*********************************************************************

MOTION_DELAY

      MOVLW H'64'
```

```
      MOVWF FIRST
      MOVLW H'64'
      MOVWF SECOND

MOVING

M2
      CALL  DELAY120US
      CALL  MOTION_SAFETY
      BTFSC SAFETY_FLAG,BIT1
      RETURN
      DECFSZ      FIRST
      GOTO  M2

      DECFSZ      SECOND
      GOTO  MOVING

      RETURN


;*********************************************************************
;BUZZER
;*********************************************************************
BUZZ
BUZZ_IT
      MOVLW 0X35
      MOVWF DCOUNT
      MOVLW 0X10
      MOVWF COUNT
      BSF         PORTD,BUZ
      CALL  DELAY
      BCF         PORTD,BUZ
      CALL  DELAY
      DECFSZ      COUNT
      GOTO  BUZZ_IT
      RETURN

;*********************************************************************
      END
```

Fig. 100 Man operating Armatron Robot