One Webcam, Multiple Robots:
Controlling Experiments Through A Network

Dilruba Akther, Amanda Hersh, Rogaite Shafi

Abstract

In this project, an individual's control of other projects with a webcam is explored. Currently, one webcam per robot is required to activate one robot via the Internet. This project makes it possible to control multiple robots through the Internet using only one webcam, which views the robots performing their tasks while located on a platform. The webcam allows the individual to visualize the robots' tasks while the network allows the user to control the robots. PBASIC is used to send commands to the platform telling it to rotate (0-360 degrees) and tilt (0-45 degrees). The webcam focuses on the various projects surrounding it. Java is used to activate these projects through the Internet. This communication of information is executed through an Ethernet board, placed in the board of education alongside the webcam. Three robots were used to test this project by having the webcam rotate to each one and having an Internet user activate them. The robots were successfully able to execute the commands given through the Internet and were viewed on the computer. This scheme is useful to businesses such as Microsoft and everyday people. They would only need to purchase and program one webcam, thereby saving time and money. It also opens up a new world of controlling operations through a network. The elderly and/or disabled individuals would no longer need to depend on others for help and would be able to retain their independence by using this project to perform everyday tasks through the Internet.

Introduction

Robots, defined by Merriam-Webster Dictionary as devices that automatically perform complicated often repetitive tasks, are utilized all over the world for a variety of tasks. Robots are able to work more efficiently than humans and are more reliable at completing a task error free. The automobile industry uses robots for painting and assembling cars. Robots can be used in dangerous jobs such as the handling of hazardous and toxic materials. Robots are capable of assisting doctors in surgery due to their precision and ability to achieve a smaller incision. Additionally, robots can be sent into areas that humans cannot investigate, like active volcanoes and far away planets.[1]

Teleoperation, also known as tele-robotics, is the control of a robot via the Internet. Human-robot interaction via the Internet is becoming more and more popular in modern society. One type of a robotic assembly planning system is called WebROBOT which

---

[1] Charles C. Weems. Computer Science. 2004.
<http://encarta.msn.com/enyclopedia_761563863_2/Computer_Science.html>.

specifies assembly jobs as well as controls a robot through the Internet.[2] A healing robot for the elderly has been constructed which is able to send and receive information through the Internet as it has been programmed to recognize twenty commonly used words. [3] Another example of teleoperation is seen in a robot built capable of receiving orders from a building management system through a LAN (local area network). This robot can be used as a surveillance robot in buildings in order to minimize the amount of people working to manage the surveillance.[4] Robots are also capable of asking a person questions if it needed help. This allows the robot to work autonomously but if it did come across any problems, it would inform the person over the Internet and the individual would be able to assist the robot. [5] Teleportation can also be used when doctors are performing surgery, with computer-based robots assisting in surgery while being controlled by a physician. These types of robots have proven to be accurate and precise. [6] Teleoperation has many unique applications. It has the potential to activate of a series of robots using a board of education, or a circuit board. The board of education consists of a breadboard for the circuitry and a basic stamp or micro controller and is connected to the Internet. The board of education is able to complete a programmed task, such as activating robots to perform tasks that are too dangerous for humans to perform; exploring active volcanoes or searching for explosives in either the same location or multiple locations. It can also activate different robots to perform different tasks in emergency situation. For example, it might trigger one robot to remove debris and rubble while setting in motion another robot to send a signal that a person has been located. A series of robots can also be activated to locate people trapped during disasters, activating multiple robots to search for multiple people.

Teleoperation is also able to aid a person who is not mobile or who does not have full manual dexterity. Such an individual may not be capable of standing up to turn the lights on and off or may not be capable of maneuvering their fingers to shift the light switch. They will be able to click on the task on their computer and activate the electricity. Similarly, if this person necessitated a change in room temperature and was not physically capable of getting to the thermostat they would be able to activate a change in the temperature of their location through the computer. Currently, it is possible for a single webcam to view a single robot which is being activated via the Internet. This project was designed to allow a single webcam to rotate and face a series of robots when activated by a person on their personal computer and thereby set in motion each robot, one at a time, to perform its specified task. The webcam is positioned on a platform in the center of the room with various robots in a circle around the webcam. An individual is able to be in a different room on the pre-designed website and select which robot they

[2] V.B Sunil and S. S. Pande. WebROBOT: Internet based robotic assembly planning system. Computers in Industry. 54:2 (2004) 191-207

[3] Takashi Oyabu, Akira Okada et al. Proposition of a survey device with odor sensors for an elderly person. Sensors and Actuators B : Chemical. 96: 1-2 (2003) 239-244

[4] Albert T.P. Soa, W.L. Chan. LAN-based building maintenance and surveillance robot. Automation in Construction. 11:6 (2002) 619-627

[5] Terrance Fonga, Charles Thorpe et al. Robot Asker of questions. Robotics and Autonomous Systems. 42:3-4 (2003) 235-243

[6] P. Vendruscolo, S. Martelli. Interfaces for computer and robot assisted surgical systems. Information and Software Technology. 43:2 (2001) 87-96

wish to see perform its specified task. A DC motor, which is programmed in PBASIC, is then activated and rotates to the programmed angle, either clockwise or counterclockwise. The DC motor, being controlled by a micro dual serial motor controller, is connected to the platform and rotates the platform so that the webcam can view the specified robot. A potentiometer, which measures electromotive force, is used to calculate the angle that the DC motor has rotated. Therefore when the individual selects on their computer which robot they want to observe perform its task, the single webcam rotates to the designated robot and that robot is activated to perform its task.

Methodology

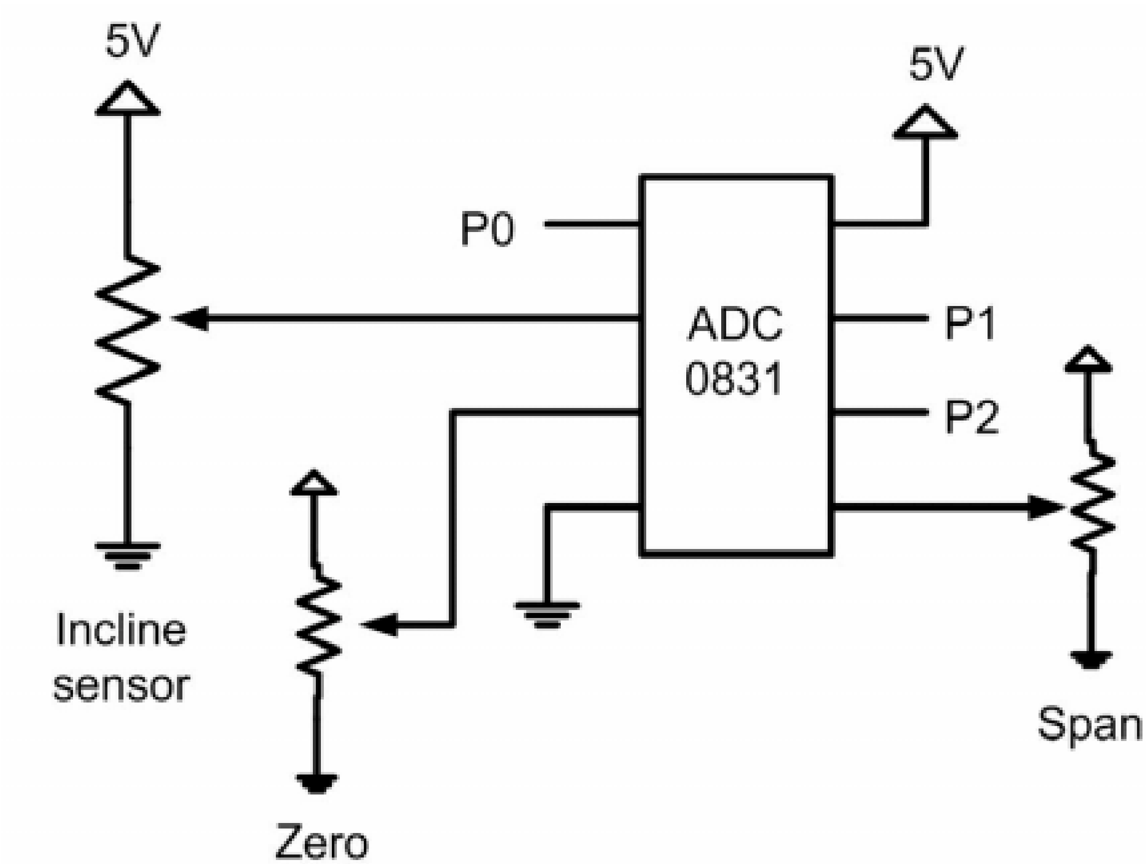
For this project, a Micro Serial Motor Controller was used to control the two DC motors that were used in this project at the same time. The motor controller was bought from Pololu. Also, the two DC motors and two potentiometers, which are used to adjust angles, were provided by the YES Center along with the jumper wires, Basic Stamp 2, and Ethernet board, which were used for the circuitry. In addition, the three platforms, the legs of the platform, the shaft, the webcam, and the hinges were provided by the YES center.

This experiment mainly consisted the use of the webcam, along with programming languages to instruct the webcam using PBASIC and Java. The first step is designing the platform for the webcam. The webcam is needed to rotate in order to focus on various experiments at once. So, a platform is added so that it can rotate and focus on the project at the same time.

The mobility of the platform is done by using hinges connecting two platforms and using a DC motor and potentiometer to move the webcam, which has been mounted on top of the platforms up or down to focus on an project. The DC motor also allowed the platform to move up or down and the potentiometer allows the platform to move at a certain angle. The two platforms were then connected to a shaft, which is in turn is connected to another platforms. This platform is mounted on four legs or stands, with the DC motor in the bottom, center of the platform, connecting to the shaft, This also contains a gear that was is joined with another gear, which is attached to another potentiometer. The DC motor enables the gears to turn while the potentiometer rotates the shaft at a certain angle.

The next step was connecting the basic stamp to the platform in order for it to rotate and programming the projects to perform their function when the webcam focuses on them using PBASIC and also placing it on the network using an Ethernet board. The last step taken was to program Java in order for a person to access these projects and control it through the net from anywhere in the world.

Schematic



Programming Code

```
'{$STAMP BS2p}
'{$PBASIC 2.5}
'CONNECTIONS FROM THE BASIC STAMP TO THE EMBEDDED ETHERNET
BOARD
' p0-p3   Crystal address bus a0-a3
' p4      /RD
' p5      /WR
' p6      AEN
' p7      N/C
```

```
' p8-p15  Crystal data bus d0-d7
' A0-A2   DAC serial connection bus
' A14-A16 ADC serial connection bus
' A7      Power amplifier pin
'
' See http://www.vermontlife.com/gary/crystal.html for information on the
Embedded Ethernet Board
' See http://www.crystal/pubs/ftp/pubs/8900.pdf for information on the
CS8900A

'
' Crystal CS8900 PacketPage equates
'
portRxTxData    CON  $00           'Receive/Transmit data (port 0)
portTxCmd       CON  $04           'Transmit Commnad
portTxLength    CON  $06           'Transmit Length
portPtr         CON  $0a           'PacketPage pointer
portData        CON  $0c           'PacketPage data (port 0)

'
' CS8900 PacketPage Offsets
'

ppProdID        CON  $0002         'Product ID Number
ppIOBase        CON  $0020         'I/O Base Address
ppIntNum        CON  $0022         'Interrupt number (0,1,2, or 3)
ppMemBase       CON  $002C         'Memory Base address register (20 bit)
ppRxCfg         CON  $0102         'Receiver Configuration
ppRxCtl         CON  $0104         'Receiver Control
ppTxCfg         CON  $0106         'Transmit Configuration
ppBufCfg        CON  $010A         'Buffer Configuration
ppLineCtl       CON  $0112         'Line Control
ppSelfCtl       CON  $0114         'Self Control
ppBusCtl        CON  $0116         'Bus Control

ppISQ           CON  $0120         'Interrupt status queue
ppRxEvt         CON  $0124         'Receiver Event
ppTxEvt         CON  $0128         'Transmitter Event
ppBufEvt        CON  $012C         'Buffer Event
ppRxMiss        CON  $0130         'Receiver Miss Counter
ppTxCol         CON  $0132         'Transmit Collision Counter
ppLineSt        CON  $0134         'Line Status
ppSelfSt        CON  $0136         'Self Status
ppBusSt         CON  $0138         'Bus Status
ppTxCmd         CON  $0144         'Transmit Command Request
ppTxLength      CON  $0146         'Transmit Length
```

```
ppIndAddr       CON  $0158         'Individual Address (IA)
ppRxStat        CON  $0400         'Receive Status
ppRxLength      CON  $0402         'Receive Length
ppRxFrame       CON  $0404         'Receive Frame Location
ppTxFrame       CON  $0A00         'Transmit Frame Location
'
' Register Numbers
'
REG_NUM_MASK      CON    $003F
REG_NUM_RX_EVENT   CON    $0004
REG_NUM_TX_EVENT   CON    $0008
REG_NUM_BUF_EVENT  CON    $000C
REG_NUM_RX_MISS    CON    $0010
REG_NUM_TX_COL     CON    $0012
'
' Self Control Register
'
SELF_CTL_RESET     CON    $0040
SELF_CTL_HC1E      CON    $2000
SELF_CTL_HCB1      CON    $8000
'
' Self Status Register
'
SELF_ST_INIT_DONE  CON    $0080
SELF_ST_SI_BUSY    CON    $0100
SELF_ST_EEP_PRES   CON    $0200
SELF_ST_EEP_OK     CON    $0400
SELF_ST_EL_PRES    CON    $0800
'
' Bus Control Register
'
BUS_CTL_USE_SA     CON    $0200
BUS_CTL_MEM_MODE   CON    $0400
BUS_CTL_IOCHRDY    CON    $1000
BUS_CTL_INT_ENBL   CON    $8000
'
' Bus Status Register
'
BUS_ST_TX_BID_ERR  CON    $0080
BUS_ST_RDY4TXNOW   CON    $0100
'
' Line Control Register
'
LINE_CTL_RX_ON     CON    $0040
LINE_CTL_TX_ON     CON    $0080
LINE_CTL_AUI_ONLY  CON    $0100
```

```
LINE_CTL_10BASET   CON    $0000
'
' Test Control Register
'


'
' Receiver Configuration Register
'
RX_CFG_SKIP        CON    $0040
RX_CFG_RX_OK_IE    CON    $0100
RX_CFG_CRC_ERR_IE  CON    $1000
RX_CFG_RUNT_IE     CON    $2000
RX_CFG_X_DATA_IE   CON    $4000
'
' Receiver Event Register
'
RX_EVENT_RX_OK     CON    $0100
RX_EVENT_IND_ADDR  CON    $0400
RX_EVENT_BCAST     CON    $0800
RX_EVENT_CRC_ERR   CON    $1000
RX_EVENT_RUNT      CON    $2000
RX_EVENT_X_DATA    CON    $4000
'
'Receiver Control Register
'
RX_CTL_PROMISCUOUS CON    $0080
RX_CTL_RX_OK_A     CON    $0100
RX_CTL_MCAST_A     CON    $0200
RX_CTL_IND_A       CON    $0400
RX_CTL_BCAST_A     CON    $0800
RX_CTL_CRC_ERR_A   CON    $1000
RX_CTL_RUNT_A      CON    $2000
RX_CTL_X_DATA_A    CON    $4000
'
'Transmit Configuration Register
'
TX_CFG_LOSS_CRS_IE CON    $0040
TX_CFG_SQE_ERR_IE  CON    $0080
TX_CFG_TX_OK_IE    CON    $0100
TX_CFG_OUT_WIN_IE  CON    $0200
TX_CFG_JABBER_IE   CON    $0400
TX_CFG_16_COLL_IE  CON    $8000
TX_CFG_ALL_IE      CON    $8FC0
'
'Transmit Event Register
'
```

```
TX_EVENT_TX_OK     CON    $0100
TX_EVENT_OUT_WIN   CON    $0200
TX_EVENT_JABBER    CON    $0400
TX_EVENT_16_COLL   CON    $1000
'
' Transmit Command Register
'
TX_CMD_START_5     CON    $0000
TX_CMD_START_381   CON    $0080
TX_CMD_START_1021  CON    $0040
TX_CMD_START_ALL   CON    $00C0
TX_CMD_FORCE       CON    $0100
TX_CMD_ONE_COLL    CON    $0200
TX_CMD_NO_CRC      CON    $1000
TX_CMD_NO_PAD      CON    $2000
'
'Buffer Configuration Register
'
BUF_CFG_SW_INT     CON    $0040
BUF_CFG_RDY4TX_IE  CON    $0100
BUF_CFG_TX_UNDR_IE CON    $0200


'
' The IP address and MAC address can be changed to whatever is appropriate
'
IP1         CON         128       'first octet of IP address
IP2         CON         238       'second octet of IP address
IP3         CON         129       'third octet of IP address
IP4         CON         91        'fourth octet of IP address

MAC1         CON         $00       '\
MAC2         CON         $00       '  \
MAC3         CON         $00       '    \
MAC4         CON         $12       '    / 48 bit IEEE OUI
(Organizationally Unique Identifier)
MAC5         CON         $34       '  /
MAC6         CON         $55       '/

rd            CON          4         ' Pin 4 -> EEB read command
wr            CON          5         ' Pin 5 -> EEB write command
aen           CON          6         ' Pin 6 -> EEB
power_pin      CON          7         ' Pin 7 -> Power on/off

addrBusOut     VAR          OUTA       ' Address Bus
dataBusIn      VAR          INH        ' Data IN Bus
```

```
dataBusOut     VAR           OUTH       ' Data OUT Bus
addr           VAR           Nib        ' Address Nib
'counter       VAR           Word       ' Counter for main loop
i              VAR           Nib        ' Counter in for loop
k              VAR           Nib        ' Counter in for loop
value          VAR           Byte
packetType     VAR           Word


' ---- { Temporary storage word } ----
dataW          VAR           Word
dataH          VAR           dataW.HIGHBYTE
dataL          VAR           dataW.LOWBYTE

offsetW        VAR           Word
offsetH        VAR           offsetW.HIGHBYTE
offsetL        VAR           offsetW.LOWBYTE
choose         VAR      Bit


' ---- { srcMAC 1--3 locations } ----
srcMAC1_H      CON           0
srcMAC1_L      CON           1
srcMAC2_H      CON           2
srcMAC2_L      CON           3
srcMAC3_H      CON           4
srcMAC3_L      CON           5


' ---- { Packet Size location } ----
length_H       CON           6
length_L       CON           7

' ---- { Control Gain memory locations } ----
'P_gain        CON           8
'D_gain        CON           9

' ---- { Data Buff locations } ----
data_buffer    CON           10


' ---- { srcIP Variables [Necessary for checksum computation] } ----
srcIP1         VAR           Word
srcIP1H        VAR           srcIP1.HIGHBYTE
srcIP1L        VAR           srcIP1.LOWBYTE
```

```
srcIP2        VAR          Word
srcIP2H       VAR          srcIP2.HIGHBYTE
srcIP2L       VAR          srcIP2.LOWBYTE


'---- { Analog to digital and digital to analog variables and constants }
----
ADres         VAR          Word           'A-to-D result: one byte.
ADresH        VAR          ADres.HIGHBYTE
ADresL        VAR          ADres.LOWBYTE
'ADres1       VAR          Word           'A-to-D result: one byte.

adcbits                VAR      Byte
angle                  VAR      Byte
previous_position      VAR   Byte
previous_position = 1




ADresNib0     VAR          ADresL.LOWNIB
ADresNib1     VAR          ADresL.HIGHNIB
ADresNib2     VAR          ADresH.LOWNIB
ADresNib3     VAR          ADresH.HIGHNIB

ADconfig      CON          %10000001       ' Configuration for
Potentiometer
ADconfig2     CON          %10010001       ' Configuration for Tachometer


AD_CS         CON          14           'Chip select is pin 14.
AD_Data       CON          13           'ADC data output is pin 13.
AD_CLK        CON          15           'Clock is pin 15.
AD_Dout       CON          12           'ADC data input is pin 12

DA_CS         CON          2           'Chip select is pin 2.
DA_CLK        CON          0           'Clock is pin 0.
DA_DATAOUT    CON          1           'input to DAC is pin 1.

theta         VAR          Word
temp          VAR          Word
CCPP          VAR          Word
```

```
number        VAR          Byte
base          CON          10


'---- {Start of the Program} ----
start:
    HIGH   rd
    HIGH   wr
    HIGH   aen
    DIRH = 0           ' data bus initially input
    DIRA = %1111       ' address bus is always output

    GOSUB verChip
    GOSUB resetChip
    GOSUB initChip
    'DEBUG "Init",CR
    'GOSUB reset_counter


read_loop:

    MAINIO
    offsetW = ppRxEvt
    GOSUB readPP   'read the receiver event
    'CCPP = CCPP + 1

    'IF (CCPP >= 3000) THEN turn_off   'This sequence makes sure that
after 1000 times of read_loop, we turn of power amp.
    IF dataH.BIT0 = 0 THEN read_loop

    'it's important to read the following data high byte first
    addr = portRxTxData+1
    GOSUB ioRead           'read and discard status
    addr = portRxTxData
    GOSUB ioRead
    addr = portRxTxData+1   'read and save length in lengthW
    GOSUB ioRead
    'lengthH = value
    PUT length_H, value

    addr = portRxTxData
    GOSUB ioRead
    'lengthL = value
    PUT length_L, value

    GOSUB recvWord
```

```
    'srcMAC1W = dataW
    PUT srcMAC1_H, dataW.HIGHBYTE              ' put in EEPROM
    PUT srcMAC1_L, dataW.LOWBYTE               ' put in EEPROM

    GOSUB recvWord
    'srcMAC2W = dataW
    PUT srcMAC2_H, dataW.HIGHBYTE              ' put in EEPROM
    PUT srcMAC2_L, dataW.LOWBYTE               ' put in EEPROM

    GOSUB recvWord
    'srcMAC3W = dataW   ' read dest MAC
    PUT srcMAC3_H, dataW.HIGHBYTE              ' put in EEPROM
    PUT srcMAC3_L, dataW.LOWBYTE               ' put in EEPROM

    GOSUB recvWord
    'srcMAC1W = dataW
    PUT srcMAC1_H, dataW.HIGHBYTE              ' put in EEPROM
    PUT srcMAC1_L, dataW.LOWBYTE               ' put in EEPROM

    GOSUB recvWord
    'srcMAC2W = dataW
    PUT srcMAC2_H, dataW.HIGHBYTE              ' put in EEPROM
    PUT srcMAC2_L, dataW.LOWBYTE               ' put in EEPROM

    GOSUB recvWord
    'srcMAC3W = dataW   ' read and save source MAC
    PUT srcMAC3_H, dataW.HIGHBYTE              ' put in EEPROM
    PUT srcMAC3_L, dataW.LOWBYTE               ' put in EEPROM

    GOSUB recvWord   'read the packet type
    packetType = dataW

    IF packetType <> $0806 THEN otherType

' This is optional code, I put this in to show how to transmit data. The
following responds TO the ARP (Address
' Resolution Protocol) request. Someone want to convert an IP address to a
MAC destination. We'll check to see
' if the request is valid and if it's for our IP address (192.168.1.2). If
so, we send the ARP response along
' with our hardware (MAC) address stored in the constants MAC1 - MAC6

    GOSUB recvWord   ' next is ar_hwtype (hardware type)
    IF dataW <> 1 THEN discardAndContinue

    GOSUB recvWord   ' next is ar_prtype (protocol type)
```

```
    IF dataW <> $0800 THEN discardAndContinue

    GOSUB recvWord    ' next is ar_hwlen (hardware address) AND ar_prlen
(protocol address length)
    IF dataH <> 6 THEN discardAndContinue
    IF dataL <> 4 THEN discardAndContinue

    GOSUB recvWord   ' next is ar_op (ARP operation 1=request, 2=reply)
    IF dataW <> 1 THEN discardAndContinue

    GOSUB recvWord   ' next is senders hardware address (ar_sha)
    GOSUB recvWord
    GOSUB recvWord

    GOSUB recvWord   ' next is senders IP address (ar_spa)
    srcIP1 = dataW
    GOSUB recvWord
    srcIP2 = dataW

    'following this is ar_tha and ar_tpa (target mac and IP). We don't
care about this since we already know who we are
    GOSUB   dropFrame ' drop the rest

    GOSUB startTx    ' start the transmission
    dataW = 42       ' length of arp is always 42, the board will pad the
runt out
    GOSUB setTxLen

waitTx:
    offsetW = ppBusSt   ' get bus status
    GOSUB readPP
    IF dataH.BIT0 = 0 THEN waitTx ' is BUS_ST_RDY4TXNOW (ready for
transmit)

    '1st, send the dest MAC address taken from the src in the arp request
    'dataW = srcMAC1W
    GET srcMAC1_H, dataW.HIGHBYTE
    GET srcMAC1_L, dataW.LOWBYTE

    GOSUB sendWord
    'dataW = srcMAC2W
    GET srcMAC2_H, dataW.HIGHBYTE
    GET srcMAC2_L, dataW.LOWBYTE

    GOSUB sendWord
    'dataW = srcMAC3W
```

```
GET srcMAC3_H, dataW.HIGHBYTE
GET srcMAC3_L, dataW.LOWBYTE

GOSUB sendWord

dataW = MAC1<<8|MAC2   'now, send our MAC address
GOSUB sendWord
dataW = MAC3<<8|MAC4
GOSUB sendWord
dataW = MAC5<<8|MAC6
GOSUB sendWord

dataW = $0806      'packet type = 0806, ARP
GOSUB sendWord

dataW = 1          'ar_hwtype = 1
GOSUB sendWord

dataW = $0800      'ar_prtype = $0800
GOSUB sendWord

dataW = $0604      'ar_hwlen = 6, ar_prlen = 4
GOSUB sendWord

dataW = 2          'ar_op = 2 (response)
GOSUB sendWord

dataW = MAC1<<8|MAC2      'ar_sha
GOSUB sendWord
dataW = MAC3<<8|MAC4
GOSUB sendWord
dataW = MAC5<<8|MAC6
GOSUB sendWord

dataW = IP1<<8|IP2        'ar_spa
GOSUB sendWord
dataW = IP3<<8|IP4
GOSUB sendWord

'dataW = srcMAC1W   'ar_tha
GET srcMAC1_H, dataW.HIGHBYTE
GET srcMAC1_L, dataW.LOWBYTE

GOSUB sendWord
'dataW = srcMAC2W
GET srcMAC2_H, dataW.HIGHBYTE
```

```
    GET srcMAC2_L, dataW.LOWBYTE

    GOSUB sendWord
    'dataW = srcMAC3W
    GET srcMAC3_H, dataW.HIGHBYTE
    GET srcMAC3_L, dataW.LOWBYTE

    GOSUB sendWord

    dataW = srcIP1          'ar_tpa

    GOSUB sendWord
    dataW = srcIP2

    GOSUB sendWord

    'DEBUG "ARP sent",CR
    GOTO read_loop

otherType:
    IF packetType <> $0800 THEN discardAndContinue      'filter only IP
packets



'---- {Decompose the IP header} ----
    GOSUB recvWord   'get ip_verlen and ip_tos
    'DEBUG "IP Ver ", DEC dataH.HIGHNIB, ", HDR Length=",DEC
dataH.LOWNIB*4,", TOS=$",HEX2 dataL,CR

    GOSUB recvWord    'get packet length
    'debug "Packet Length=",   dec dataW,cr

    GOSUB recvWord   'ip_id
    'debug "Datagram ID=$", hex4 dataW,cr

    GOSUB recvWord   'ip_fragoff
    'debug "Frag Offset=",dec dataW,cr

    GOSUB recvWord   'ip_ttl & ip_proto
    'debug "TTL=",dec dataH,cr

    IF dataL <> 17 THEN notUDP
    'debug "Protocol=UDP",CR
    GOTO nextHdrField
```

```
notUDP:
    GOTO discardAndContinue

nextHdrField:
    GOSUB recvWord   'ip_cksum
    'debug "Checksum=$",hex4 dataW,cr

    'debug "Src IP Address="
    GOSUB dumpIP2

    'debug "Dest IP Address="
    GOSUB dumpIP


    'lengthW = lengthW - 34 / 2 'subtract the 2 MAC (3 words each) and the
protocol type (2 bytes) AND the 20 Byte header

    ' dump out the packet data.
    'DEBUG "Packet Data:",CR

    GOSUB recvWord
    'DEBUG ? dataW
    IF dataW = $03E8 THEN next_check
    GOTO read_loop

next_check:
    FOR i = 0 TO 3          'WE MODIFIED THIS to be 0 to 4 instead of 0 to
lengthW
       GOSUB recvWord

checkend:
    NEXT
    IF dataW = $6F6F THEN move_motor  '138 is the key, user has to send
this in first DATA Byte

    GOTO read_loop

move_motor:
    'AUXIO
'      HIGH power_pin
'      PAUSE 100
    'MAINIO

    GOSUB recvWord
    packetType.LOWBYTE = dataL
    packetType.HIGHBYTE = dataH
```

```
    'IF packetType = $FFFF THEN reset_counter
    'theta = packetType
    GOSUB recvWord
    'DEBUG CR, DEC dataH, "--", DEC dataL, CR  '  This recvWord gets Pgain
    'PUT P_gain, dataH                ' put in EEPROM
    angle = dataH
    'PUT D_gain, dataL                ' put in EEPROM

   '---- { PD CONTROL ALGORITHM } ----
   AUXIO

initial:

GOSUB stop_moving


'DEBUG ? angle

'HIGH 0
'LOW 0
'LOW 1
'PULSOUT 1, 210
'SHIFTIN 2, 1, MSBPOST, [adcbits\8]

prepare:
GOSUB stop_moving
'DEBUG CLS
'DEBUG "select position from 1-3"   ,CR
'DEBUGIN  DEC angle



main:


GOSUB check_top_ADC
DEBUG ? adcbits

number = 1

GOSUB moving_down


   '---- { END OF PD CONTROL ALGORITHM } ----
```

```
   MAINIO
   GOTO sending_packet

'----{ SENDING OF PACKET }----
sending_packet:
    'counter = counter + 1
    CCPP = 1

'Old way of computing packetType
'      packetType = $8679 + srcIP1 + srcIP2 + 3 'packetType variable is
being recycled again FOR holding the checksum
'      packetType = $FFFF - packetType

'New way:
    'Calculate checksum
    packetType = $FFFF - $8679 - srcIP1 - srcIP2 - 3 'packetType variable
is being recycled again FOR holding the checksum


    GOSUB startTx       ' start the transmission
    dataW = $0030       ' Length of entire transmission including link
layer DATA (bytes)
    GOSUB setTxLen

waitTy:
    offsetW = ppBusSt      ' get bus status
    GOSUB readPP
    IF dataH.BIT0 = 0 THEN waitTy    ' is BUS_ST_RDY4TXNOW (ready for
transmit)


    '1st, send the dest MAC address taken from the src in the arp request
    'dataW = srcMAC1W
    GET srcMAC1_H, dataW.HIGHBYTE
    GET srcMAC1_L, dataW.LOWBYTE

    GOSUB sendWord
    'dataW = srcMAC2W
    GET srcMAC2_H, dataW.HIGHBYTE
    GET srcMAC2_L, dataW.LOWBYTE

    GOSUB sendWord
    'dataW = srcMAC3W
    GET srcMAC3_H, dataW.HIGHBYTE
```

```
    GET srcMAC3_L, dataW.LOWBYTE

    GOSUB sendWord

    dataW = MAC1<<8|MAC2   'now, send our MAC address
    GOSUB sendWord
    dataW = MAC3<<8|MAC4
    GOSUB sendWord
    dataW = MAC5<<8|MAC6
    GOSUB sendWord

    dataW = $0800      'packet type = 0800, IP
    GOSUB sendWord


    '********* end of ether

    dataW = $4500      '** IP Version/Header Length (32bit words)
    GOSUB sendWord

    dataW = $0022      '*** TOTAL Packet Length (bytes) ***

    GOSUB sendWord

    dataW = $0000      '*** ID (if fragment) ***
    GOSUB sendWord

    dataW = $4000      '***** FLAGS/OFFSET *****
    GOSUB sendWord

    dataW = $FF11      '***** TTL/PROTOCOL *****
    GOSUB sendWord

    dataW = packetType '******* IP CHKSUM *******   'MUST BE CHANGED for
each different packet OR will be droped
    GOSUB sendWord

    dataW = $80EE      '******* SRC IP 1 ********
    GOSUB sendWord

    dataW = $8158      '******* SRC IP 2 ********
    GOSUB sendWord

    dataW = srcIP1     '******* DEST IP 1 *******
    GOSUB sendWord
```

```
    dataW = srcIP2     '******* DEST IP 2 *******
    GOSUB sendWord


    dataW = $03E8      '******** SRC PORT *******
    GOSUB sendWord

    dataW = $03E8      '******* DEST PORT *******
    GOSUB sendWord

    dataW = $000E      '******** LENGTH *********
    GOSUB sendWord

    dataW = $0000      '***** UDP CHECKSUM ******
    GOSUB sendWord

    'dataH = ADres     'CHANGED FROM Error
    'dataL = ADres     '********* DATA (position reading) **********

    FOR k = 0 TO 1
      GET data_buffer+(2*k), dataW.HIGHBYTE
      GET data_buffer+(2*k)+1, dataW.LOWBYTE
      GOSUB sendWord
    NEXT

    'dataW = counter        '********* DATA (sample number) **********
    dataW = 1         '********* DATA (sample number) **********
    GOSUB sendWord


'------------- END of SENDING OF PACKET -----------------

  GOTO read_loop

dumpIP:
  GOSUB recvWord
  'DEBUG DEC dataH,".",DEC dataL,"."

  GOSUB recvWord
  'DEBUG DEC dataH,".",DEC dataL,CR
  RETURN

dumpIP2:
  GOSUB recvWord

  srcIP1H = dataH
```

```
  srcIP1L = dataL
  'DEBUG DEC dataH,".",DEC dataL,"."

  GOSUB recvWord
  srcIP2H = dataH
  srcIP2L = dataL
  'DEBUG DEC dataH,".",DEC dataL,CR

  RETURN

discardAndContinue:

  GOSUB dropFrame
  GOTO read_loop


recvWord:
  addr = portRxTxData
  GOSUB ioRead
  dataH = value
  addr = portRxTxData+1
  GOSUB ioRead
  dataL = value
  RETURN

' Sends the transmit start command to the board
startTx:
  dataW = TX_CMD_START_ALL
  addr = portTxCmd
  value = dataL
  GOSUB ioWrite
  addr = portTxCmd+1
  value = dataH
  GOSUB ioWrite
  RETURN

' Sends the length of the transmission contained in dataW
setTxLen:   value = dataL
  addr = portTxLength
  GOSUB ioWrite
  value = dataH
  addr = portTxLength+1
  GOSUB ioWrite
  RETURN

' Transmits the word at dataW
```

```
sendWord:
   addr = portRxTxData
   value = dataH
   GOSUB ioWrite
   addr = portRxTxData+1
   value = dataL
   GOSUB ioWrite
   RETURN

dropFrame:
   offsetW = ppRxCtl
   GOSUB readPP
   dataW = dataW | RX_CFG_SKIP
   GOSUB writePP
   RETURN


' ---- {Initializes the CS8900} ----
initChip:
   offsetW = ppLineCtl
   dataW = LINE_CTL_10BASET
   GOSUB writePP          ' set to 10BaseT
   offsetW = $0118        'ppTestCtl con $0118 'Test Control
   dataW = $4000
   GOSUB writePP          ' set to full duplex
   'no   offsetW = ppRxCfg
   'irqs   dataW = RX_CFG_RX_OK_IE
   '      gosub writePP
   offsetW = ppRxCtl
   dataW = RX_CTL_RX_OK_A|RX_CTL_PROMISCUOUS
   GOSUB writePP
   'no   offsetW = ppTxCfg
   'irqs   dataW = TX_CFG_ALL_IE
   '      gosub writePP
'
' Important: The IA needs to be byte revered IA=aa:bb:cc:dd:ee:ff
'
   offsetW = ppIndAddr
   dataW = MAC2<<8|MAC1
   GOSUB writePP
   offsetW = ppIndAddr+2
   dataW = MAC4<<8|MAC3
   GOSUB writePP
   offsetW = ppIndAddr+4
   dataW = MAC6<<8|MAC5
   GOSUB writePP
```

```
   '      offsetW = ppBusCtl
   'no   gosub readPP
   'irqs   dataH.bit7 = 1       ' enable irq
   '      gosub writePP
   offsetW = ppLineCtl    ' get line control
    GOSUB readPP
   dataL.BIT6 = 1       ' SerRxOn
   dataL.BIT7 = 1       ' SerTxOn
   GOSUB writePP
   RETURN



' ---- {Resets the CS8900 and checks to insure initialization done bit is
set} ----
resetChip:
   offsetW = ppSelfCtl
   dataW = SELF_CTL_RESET
   GOSUB writePP      ' issue a reset to the chip


resetWait:
   PAUSE 1         ' wait 1 millisecond
   offsetW = ppSelfCtl    'get the Self Control status
   GOSUB readPP
   'debug "ppSelfCtl=",HEX4 dataW,cr
   IF dataL.BIT6 = 1 THEN resetWait
            ' bit 6 cleared, chip is reset
   offsetW = ppSelfSt    'get self status
   GOSUB readPP
   'debug "ppSelfSt=",HEX4 dataW,cr
   IF dataL.BIT7 = 0 THEN resetWait    ' INITD means initialization is done
when set
   'debug "CS8900 RESET",cr
   RETURN

verChip:                  ' first, get the signature at portPtr which should
be $3x0x
   addr = portPtr

   GOSUB ioRead
   dataL = value
   addr = portPtr+1
   GOSUB ioRead
   dataH = value
   IF dataH.HIGHNIB = 3 THEN validChip
```

```
  END

validChip:
  'DEBUG "Signature=", HEX4 dataW,CR
  offsetW = $0000                   'ppEISA con $0000 'EISA
Registration number of CS8900
  GOSUB readPP
  'DEBUG "EISA=", HEX4 dataW,CR
  offsetW = ppProdID
  GOSUB readPP
  'DEBUG "ProdID=", HEX4 dataW,CR
  RETURN

' ---- {Writes the value at dataW to the packet page register at offsetW}
----
writePP:
  GOSUB setPPPointer
  addr = portData
  value = dataL
  GOSUB ioWrite
  addr = portData+1
  value = dataH
  GOSUB ioWrite
  RETURN


' ---- {Read packet page data at offsetW and put result in dataW} ----
readPP:
  GOSUB setPPPointer
  addr = portData
  GOSUB ioRead
  dataL = value
  addr = portData+1
  GOSUB ioRead
  dataH = value
  RETURN

' ---- {Sets the packetpage address} ----
setPPPointer:
  value = offsetL
  addr = portPtr
  GOSUB ioWrite
  value = offsetH
  addr = portPtr+1
  GOSUB ioWrite
  RETURN
```

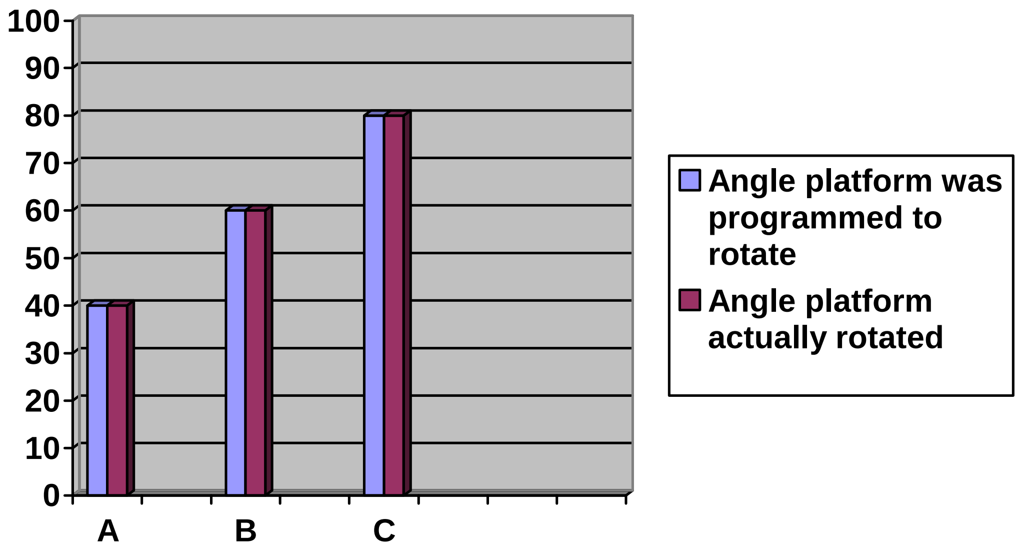
```
ioRead:
  DIRH = 0              ' make data bus input
  addrBusOut = addr
  LOW aen
  LOW rd
  value = dataBusIn
  HIGH rd
  HIGH aen
  RETURN

ioWrite:
  DIRH = %11111111      ' make data bus output
  dataBusOut = value
  addrBusOut = addr
  LOW aen
  LOW wr
  HIGH wr
  HIGH aen
  RETURN



'reset_counter:
  'counter = 1
'   CCPP = 1
'   RETURN



'-------- New Subroutines ---------


initialize_motor:
HIGH 4
LOW 5
HIGH 5
PAUSE 10
RETURN


moving_down:
GOSUB  check_top_ADC
IF (number -10) <= adcbits AND (number +10) >= adcbits THEN  RETURN
```

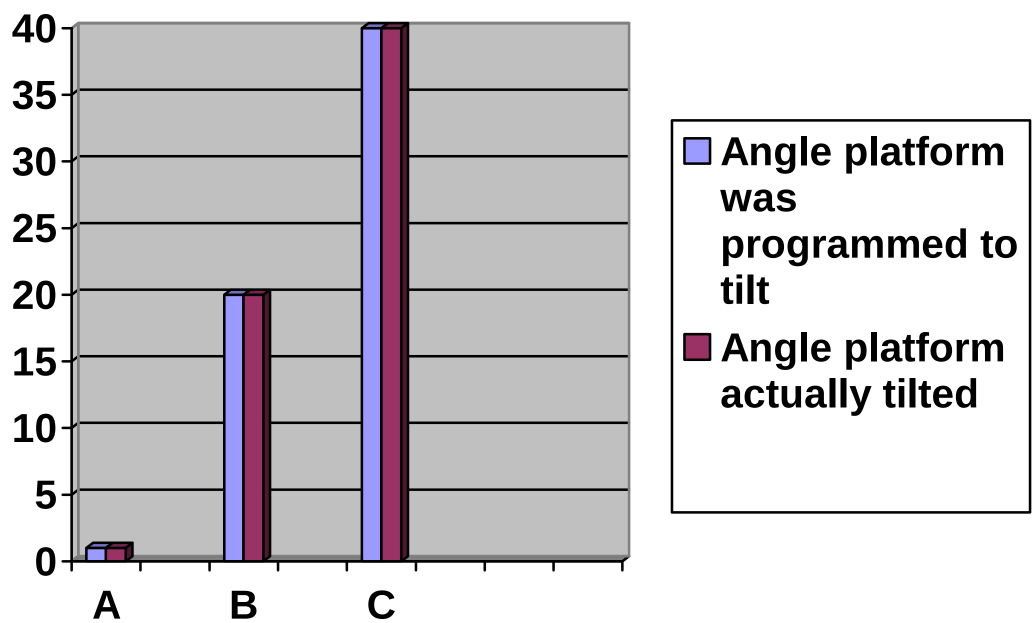
```
GOSUB initialize_motor
SEROUT 4,240,[$80,0,3,44]
PAUSE 20

GOTO moving_down


moving_upward:
GOSUB  check_top_ADC

IF (number -10) <= adcbits AND (number +10) >= adcbits  THEN  RETURN

GOSUB initialize_motor
SEROUT 4,240,[$80,0,2,60]
PAUSE 20

GOTO moving_upward




moving_clockwise:
GOSUB check_bot_ADC

IF (theta -2) <= adcbits AND (theta +2) >= adcbits THEN RETURN

GOSUB initialize_motor
SEROUT 4,240,[$80,0,0,66]          'shaft rotates clockwise
PAUSE 20

GOTO moving_clockwise



moving_counterclock:
GOSUB  check_bot_ADC

IF (theta -2) <= adcbits AND (theta +2) >= adcbits THEN RETURN

GOSUB initialize_motor
SEROUT 4,240,[$80,0,1,55]
PAUSE 20

GOTO moving_counterclock
```

```
check_top_ADC:
'DEBUG CLS
HIGH 0
LOW 0
LOW 1
PULSOUT 1, 210
SHIFTIN 2, 1, MSBPOST, [adcbits\8]
'DEBUG CR, "Value for Top ADC   ", ? adcbits
RETURN

check_bot_ADC:
'DEBUG CLS
HIGH 10
LOW 10
LOW 11
PULSOUT 11, 210
SHIFTIN 12, 11, MSBPOST, [adcbits\8]
'DEBUG CR, "Value FOR Bottom ADC   ", ? adcbits
RETURN


stop_moving:
GOSUB initialize_motor
SEROUT 4,240,[$80,0,1,0]

GOSUB initialize_motor
SEROUT 4,240,[$80,0,3,0]
PAUSE 20
RETURN

set_angle1:
theta = 40
number = 1
RETURN

set_angle2:
theta = 60
number = 20
RETURN

set_angle3:
theta = 80
number = 40
RETURN
```

Data

	Angle platform was programmed to rotate	**Angle platform actually rotated**	**Angle platform was programmed to tilt**	**Angle platform actually tilted**
Robot A	40°	40°	1°	1°
Robot B	60°	60°	20°	20°
Robot C	80°	80°	40°	40°





Discussion

After conducting our experiment we came to the decision that the webcam would not be able to rotate to any position from 0-360 because it was originally programmed to move only clockwise. Because of this an angle moving from 270 degrees to 269 degrees would have to complete a 359-degree turn. Instead, it was decided to make three pre-programmed positions that would turn either clockwise or counterclockwise depending on where it was. If it was at position three and needed to go to position two or one it would turn counterclockwise. If it was at position one and needed to go to position two or three it would turn clockwise. If it was at position two it would turn clockwise to three and counterclockwise to one. In the future this project could be enhanced by adding temperature and light sensors and by programming it to activate other robots according to their environmental needs.

Conclusion

After conducting the experiment, it was proved that making one web camera perform the task of several robots was successful. The camera also succeeded in giving feed to the Internet, making it possible for others to activate robots from various locations without having to see the robot they are activating. Although the serial micro dual controller performed its purpose, it was suggested that it was not a dependable component because it tended to burn out several times without much cause.

References

Charles C. Weems. Computer Science. 2004. <http://encarta.msn.com/enyclopedia_761563863_2/Computer_Science.html>.

V.B Sunil and S. S. Pande. WebROBOT: Internet based robotic assembly planning system. Computers in Industry. 54:2 (2004) 191-207.

Takashi Oyabu, Akira Okada et al. Proposition of a survey device with odor sensors for an elderly person. Sensors and Actuators B : Chemical. 96: 1-2 (2003) 239-244.

Albert T.P. Soa, W.L. Chan. LAN-based building maintenance and surveillance robot. Automation in Construction. 11:6 (2002) 619-627.

Terrance Fonga, Charles Thorpe et al. Robot Asker of questions. Robotics and Autonomous Systems. 42:3-4 (2003) 235-243.

P. Vendruscolo, S. Martelli. Interfaces for computer and robot assisted surgical systems. Information and Software Technology. 43:2 (2001) 87-96.