University of Florida
Intelligent Machine Design Lab

Card Shark

(final report)

Kevin E. Kane
ID # 72860210
Dr. A. Arroyo

Table of Contents

Abstract

Card Shark is an autonomous professional card-playing robot specializing in blackjack play. Using its own deck of cards with barcodes on them the robot can place bets, read its cards and the dealer's cards, and then makes the proper mathematical play for best long run results. An excite toy for everyone to watch, Card Shark's uses of entertainment could even be used directly in casinos.

Executive Summary

In an attempt to better educate the gambling community about the simple, yet exciting game of blackjack, I've designed Card Shark. Card Sharks uses are completely for entertainment and spectator enjoyment for now. In the flashy world of Gambling and card tables, a simple robot playing a traditionally human game would be an attention grabber. Many would most likely what to even play next to the robotic gambling marvel. It does seem that future designs could also be build with more of an education/training purpose in mind. Card Sharks could be used for teaching the game to complete novices and be a fun table companion at the same time.

Card Shark is designed to function as a completely autonomous blackjack player. Card Shark has three main behavior functions that it performs, while in combination they create a false sense of intelligence. It follows a highly contrast line track counter clockwise. As it performs this task, it scans the barcodes on playing cards laid out before it. By reading and inputting these card values (or instructions as it maybe considered) Card Shark makes a blackjack playing choice. Doubling down, splitting, hitting or standing is all part of a standard repertoire for this gambling machine. In split and doubling down situations Card Shark's current design drops a colored chips to better express its desires. Complete hands are displayed on the LCD screen readable to either the dealer or observers.

With the ability to play one hand after another, and never become mentally tired, Card Shark is the first truly perfect Blackjack player. Fun to watch and possibly even learn from, Card Shark may bring an ever more positive light to the some times intimidating game of blackjack. If casinos desires are to keep patrons entertained and on the gaming floor having a couple Card Sharks in their casino may have the potential to bring more business and bets then before.

Introduction

Card Shark is the first autonomous blackjack-playing robot ever (at lest I think so). It is mostly for fun in its current form, as no serious casino would allow a robot to play its blackjack tables. However, warehouses or other shipping companies can use the basic programming level of Card Shark's decision-making and barcode reading to move organize and distribute products. With a computerized system to organize and place goods in/out of storage, a company can expect exact and dependable work.

Integrated System

At the heart of Card Shark is an ATMEL Atmega32L micro-controller mounted on a STK500 testing board. This micro controller directly interfaces with the following devices:

- 2 "hacked" servos for mobility around the playing table
- 2 "arm" servos for placing bets
- 1 LCD for feedback to the dealer and operator
- 2 photo resistor circuits for basic line following
- 1 KaneScan barcode scanner for playing card reading

There are two movement servos that are used to drive Card Shark around the table while following the black line. These two servos are used to help it turn left or right. An interesting aspect of Card Shark is how wide it is. As a result, in order to make sharp turns I wrote code that sets one wheel in reverse (although it is slow) in order to make tight turns as I desired. This makes the Card Shark's turns/movements appear jerky at times; however the advantages of a smaller track can then be utilized.

The "arms" are the most internal part of my design and are used to drop different colored chips from inside the chip holder areas. These chips are pushed to the cutout holes at the bottom of the platform. These servos are not hacked and are used to move one way and then back to their resting positions.

The LCD is used to inform the dealer and operator what Card Shark wishes to do with its decision; hit, stand, double down, or split. It is located on the top of Card Shark.

The photo resistors are used to follow a black line. They are incorporated in a circuit designed to give a digital (1 = white, 0 = black line) output. This keeps Card Shark in the proper place for reading the cards in order.

The Barcode scanner is used to recognize the actual card values. Each card has a barcode on it that is scanned and input to Card Shark as it drives by. A basic Code 39 is used for the scanner and barcodes. The nice thing about this is that numbers are sent in ASCII form. These means that after reading in the value the code could subtract $30 and have the correct hexadecimal value to be used. (Notice that Aces = $01 and Tens = $00)

Mobile Platform

The platform I designed in AutoCAD and cutout with the T-tech was quite large with dimensions of 13” x 8”. It holds the batteries and scanner on one side and the chips on the other, while still carrying the full STK500 in the middle. That is the scanner is on the left and playing cards are on the right when facing the same direction as Card Shark. The LCD is located on top next to the board and faces the back of Card Shark. This seemed to be the best location for both the operator and the dealer to see what Card Shark’s playing choice is. An area was built to hold ten AA batteries to power the robot. On top of it is velcro to hold the scanner in place during operation. In this design velcro is the only choice as glue means you’d be unable to replace Card Shark’s batteries and anything less would not hold the scanner in place. Along with the wooden frame are two separate “arms” for moving chips from their holding locations to the holes cut out in the bottom layer of the platform. A hole large enough to drop one chip bets as either the original, double down, or splitting bets. The frame is held together using screws; in this way the separate layers of the platform can be taken apart and line following circuits potentiometers may be adjusted and the betting “arms” inside can be inspected.

Actuation

Card Shark moves vie two servos, one for each driving wheel. The other two servos are independently used for the betting “arms” inside the robot’s platform. These servos are moved in three different patterns to displace three different bets; first bet, doubling down, and splitting bets. For this reasons two of the used were bought as “hacked” servos while the remained standard servos.

As a funny side note, two of my original servos (non-hacked) were destroyed when I put an excess amount of crazy glue on the dowels placed inside the rotating shafts of the servos. They were glued stiff and wouldn’t move!

Sensors

Sensor: **KaneScan Barcode Scanner** (just happens to have my last name in it)
Part number: 02001291
Qty: 1
Discussion: The KaneScan barcode scanner is an affordable ($85) highly accurate (4 mils) barcode scanner. Each card in Card Shark's deck has a barcode on the top corresponding to its value in blackjack. Aces are coded with the number 1 and are handled by the software as either 1 or 11 like in all blackjack casino games. Tens are given a value of 0 by the barcoding scheme but fixed in the programming code. The barcode scanner is mounted on the left with it scanning done behind the drive wheels. The scanner is interfaced via an RS-232 into an extra UART port built into the micro-controller.

Sensor: **Photo resistors**
Part number: none
Qty: 2
Discussion: Card Shark uses two photo resistant cells for following a black line. There have been many projects that have done similarly and I followed those designs tightly. I used the same circuit design given in a report on line tracking by William the TA. By following a black line Card Shark is sure to stay on track and read all the playing cards in proper order and then place the bets in the same general locations.

Behaviors

First Card Shark places its starting bet to be in the hand. Card Shark then follows the black line track around to scan the dealer's cards and then its own cards. It then makes the choice to hit, stand, double down, or split. Card Shark displays this decision on its LCD screen (preceded by the card's value read) and if needed drops the correct betting chip for doubling down or splitting. Card Shark then continues around the track prepared to read the next card if a hit or split was the play, or ready to read the next hand if doubling down or standing was the play.

These behaviors major components were worked on separately (line tracking, barcode reading, and betting) and then I tried to blend them together. I discovered many issues to deal with when these behaviors were brought together. One was the power needed to fully supply the STK500 board, 4 separate servos, two photo resistor circuits and one barcode scanner was a lot. I used 10 rechargeable Energizer 2300 mAh batteries and this was barely enough. (Notice: for the teacher/TA demo I tried different batteries with very bad results, including a very jumpy robot that wouldn't follow a line) Another issue with blending these behaviors is that scanning the cards is easy by hand but while driving exact alignment is needed. Slight changes in approach angles result in unread cards.

Conclusion

Card Shark is a fully functional blackjack-playing robot. I know how to read/scan cards calculate its best betting solution and drop the proper chip for that decision. It is of course only for fun and possibly for training yourself at home. However, the basic ideas behind Card Shark are far reaching. They could be used in moving and dropping off supplies (chips) to different location in a specified track or warehouse. By having interchangeable barcodes, operators (dealers) can layout different patterns or objectives for an army of robots without actually changing the robots themselves. This system would save a lot of time and effort if many such robots were in operation at a warehouse, factory, distribution center, or any other quickly changing (yet robotically feasible) manual labor locations.

There are many, many problems with this first design of Card Shark. I will explain a couple now, but many more improvements should be made in any next generation designs. First of all my original plans were to have Card Shark carry many doubling down bet and splitting bet chips with it as it moved. It was to drop one at a time and still hold a supply of extra chips for following hands. The problem was the material used in the internal "arms" was wood. Everything was laid out in AutoCAD and finely designed to coordinate tightly with one another; however, the wood was quick to warp and bend, making my dream system impossible to utilize. I was left sanding, carving, gluing and praying only to get it holding and dropping one chip at a time (I'm clearly not a mechanical engineer). Second I never clearly laid a plan to make Card Shark stop and wait/drop bets at an exact location. One black tab and a third photo-resistor would have solved this issue. Instead, I the dealer, guessed the location of the last playing card so the bet chips would stay in the same general area.

I believed the easiest part of my project would be writing the code for playing the "basic" strategy laid out in books and online. This wasn't the case and what I expect to be a couple days work turned into assembly code over 400 words in length, and even then it was far from perfect. "Arrays" and "lookup tables" coded in a higher-level language are the only ways of doing this. I tried indirect indexing but faced more problems then I was able to tackle. The result is a maze of code through branching and jump commands. (I challenge any normal human being to try and follow it)

I would say that although Card Shark does not act completely to my desires or even correctly 100% of the time, I'm still proud of the work and effort I put into it. The code is huge, all available PWMs were used, I almost ran out of ports on the STK500, and I burned, destroyed and lost more components then I'd like to admit. It was all a learning experience. The fact is, I learned more practical things in this short summer then any other class at UF.

Acknowledgements

One thing I learned in this course was not to just know the right people, but to know the right questions to ask those people! With that I'd like to thank, Steven Pickels for his extensive help with my many barcode scanner questions, William Dubel for direction with the line tracking and power supply circuits, Max Koessick for cutting out my platform and teaching me AVR assembly code. In addition I'd like to thank a couple students in the class as well, Jeffery Cohan for the photo-resistors, Stephen Corbett for sending me PWM example code, and the builder of "please-don't-suck-bot" for the servo wheels he offered me in my time of need.

Last I'd like to thank A. A. Arroyo and E. M. Schwartz for the opportunity they gave us to take this class. Thanks to their efforts and teachings the world of robotics has been opened to us all.

Appendix

Completed Code:

```
; ******************************
; **        Kevin E Kane             **
; **        7/28/2004                **
; **        EEL 5666, UF             **
; **        Card Shark 2.d           **
; *****************************************************************
; * Program Discription:  Is the second try at intergrating the blackjack        *
; * playing program of Card Shark with the actuation fuctions.  These            *
; * actuations include the servos used for driving, sensors for line             *
; * detection and additional servos for placing bets (starting, splitting        *
; * and doubling down.                                                           *
; *****************************************************************

; ***********Notes To Yourself******************
; 1)        PortA is for LCD screen!
; 2)        PortB is for anything (pin 3 is a timer PWM output, OC0)
; 3)        PortC is for LEDs (NOTE: only pins 7, 6, 1, and 0 seem to work)
; 4)        PortD is for TX(transmiter), RX(resiever), V0, V1(eyes),
;                   PWM0, PWM1 (wheels), OC2 (splitting/dd pin)
; 5)        @@@@@@ = means these lines may be removed but are used for trouble shooting

.nolist
.include "C:\Program Files\Atmel\AVR Tools\AvrAssembler\Appnotes\m32def.inc"
.list

;***** Declarations ****************************
.def      temp0             =r16      ; Temporary register 0
.def      temp1             =r17      ; Temporary register 1 (for UART I/O)
.def      temp2             =r18      ; Temporary register 2 (for LCD screen/PortA)
.def      temp3             =r19      ; Temporary register 3
.def      Delay1            =r20      ; Delay variable 1
.def      Delay2            =r21      ; Delay variable 2
.def      Delay3            =r22      ; Delay variable 3
.def      mpr               =r23      ; "Multi-Purpose Register" (used for servos)
.def      mpr2              =r24      ; "Multi-Purpose Register", number 2 (used for servos)

;***** Interrupt Vectors *******************************
.org      $000
          rjmp      Init                          ; Starting Line (jump to Init)
.org      $01A
          rjmp      Scan                          ; UART Receive Complete Interrupt Vector Address ($01A)

;*********************************************************
;***** INITIALIZATIONS!!! ******************************
Init:
;     ****Port Setups****
          ser               temp0
          out               DDRA,temp0                    ; Set PORTA to all outputs
          out               DDRB,temp0                    ; Set PORTB to all outputs
          ldi               temp0,0b10111111
          out               DDRC,temp0                    ; Set PORTC to all outputs except pin6 input
          ldi               temp0,0b10110000
          out               DDRD,temp0                    ; Set PORTD pins 4, 5 and 7 as outputs, all others are
inputs

          clr               temp0
          out               PortA,temp0                   ; Set no "pullups" for PortA
          out               PortB,temp0                   ; Set no "pullups" for PortB
          out               PortC,temp0                   ; Set no "pullups" for PortC
          ldi               temp0,0b10110000              ; Set pins 4 and 5 as high initially,...
          out               PortD,temp0                   ; ...no other "pullups" for PortD
;                   ****Enable Output Compare, 8-bit Timers (arms, mode 4)****
          ldi               mpr,0b01110100      ; (timer 0)         ; mode 1 PWM, "set" on rise and...
```

```
        out             TCCR0,mpr                                       ; ..."clear" on fall, prescaler equals 256
        ldi             mpr,$EF                                         ; compare value
        out             OCR0,mpr                                ; $EF = starting position
        ldi             mpr,0b01110110  ; (timer 2)             ; mode 1 PWM, "set" on rise and...
        out             TCCR2,mpr                                       ; ..."clear" on fall, prescaler equals 256
        ldi             mpr,$EC                         ; compare value = $EC (changed back at the end of init.)
        out             OCR2,mpr                                ; NOTE:Place first bet, before LCD delays...
                                                                        ; ...and before Card Shark starts moving
;   ****START****
polling:                                ; polling for Port C, pin6 to be pushed
        out             PORTB,temp0
        sbis    PINC,0x06                               ; If (Port C, pin6 ==0)
        rjmp    Start                                   ; then jump to "start" and finish Initializations
        inc             temp0                                   ; else inc temp0 value and...
no_start:
        dec             Delay1
        brne    no_start
        dec             Delay2
        brne    no_start
        rjmp    polling
Start:
;               ****Stack Pointer setup****
        ldi             temp0,high(Ramend)
        out             SPH,temp0
        ldi             temp0,low(Ramend)
        out             SPL,temp0                       ; Stack pointer points to end of RAM ($085F)
;               ****Drop first bet****
        ldi             mpr,$F6                                 ; compare value = $F6 (changed back at the end of init.)
        out             OCR2,mpr                                ; NOTE:Place first bet, before LCD delays...
;               ****Enable Output Compare, 16-bit Timer (Wheel PWMs, mode 8)****
        ldi             mpr,0b11110000                          ; mode 8 PWM, "set" on rise and "clear" on fall
        out             TCCR1A,mpr
        ldi             mpr,0b00010100                          ; mode 8 PWM, prescaler equals 256
        out             TCCR1B,mpr
        ldi             mpr,$01
        ldi             mpr2,$38
        out             ICR1H,mpr                                       ; Set "TOP" equal to $138
        out             ICR1L,mpr2                                      ; 20.0ms
;               ****UART setup****
        ldi             temp1,0b00000000
        out             UBRRH,temp1
        ldi             temp1,51                        ; Set UART for 9600 baud rate
        out             UBRRL,temp1
        ldi             temp1,0b10000000  ; Receive complete flag
        out             UCSRA,temp1
        ldi             temp1,0b10010000  ; Enable UART Receiver, and Receive Interrupt
        out             UCSRB,temp1
        ldi             temp1,0b10000110  ; Enable Asynchronous UART operation,
        out             UCSRC,temp1                             ; 8-bit data packs, and no parity
;               ****Prepare LCD screen****
        ldi             temp2,$00                       ; Set "enable" bit low/off
        out             PORTA,temp2
;                                                       (4-bit enable)
        ldi             temp2,$03
        call    LCDdelay
        ldi             temp2,$03
        call    LCDdelay
        ldi             temp2,$03
        call    LCDdelay
        ldi             temp2,$02
        call    LCDdelay
;                                                        (2-line enable)
        ldi             temp2,$02
        call    LCDdelay
        ldi             temp2,$08
        call    LCDdelay
;                                                        (Display, Cursor, Blink)
        ldi             temp2,$00
        call    LCDdelay
        ldi             temp2,$0F
        call    LCDdelay
;                                               (Clear Home)
        ldi             temp2,$00
        call    LCDdelay
        ldi             temp2,$01
        call    LCDdelay                        ; L.C.D. SCREEN READY!!!!
;       ldi             temp0,$99                       ;@@@@@@@@
;       out             PORTB,temp0                     ;@@@@@@@@
;               ****Set Starting Variable Values****
        ldi             temp0,$02
        sts             final_card_number,temp0
        clr             temp0
        sts             card_number,temp0
        sts             dealer_card,temp0
        sts             players_hand_count,temp0
;               ****betting arm returns to starting position****
        call    servo_delay                             ; delay longer for "arm" to finish its first move
        ldi             mpr,$ED                         ; compare value = $ED
        out             OCR2,mpr

        ldi             temp0,$1F                       ;@@@@@@@@@@@@@@
        out             PORTB, temp0                    ;@@@@@@@@@@@@@@
;               ****Interrupt setup****
        sei                                                     ; enable interrupts
;               ****Lights signaling end of Init****
;       ldi             temp0,$AA                       ;@@@@@@@@@@
;       out             PORTC,temp0                     ;@@@@@@@@@@
;**********************************************

;**********************************************
;***********START OF MAIN PROGRAM!!!!***********
;*   This program follows a line while        *
;*   staying in the "mainloop" program.       *
Mainloop: ;***Start of Mainloop program*******
        sbis    PIND,0x02                               ; is Pin number 2 of Port D (white) low?
        rjmp    turn_left                       ; if so then jump to "turn_left"
        sbis    PIND,0x03                               ; is Pin number 3 of Port D (white) low?
        rjmp    turn_right                      ; if so then jump to "turn_right"
straight:
        ldi             mpr,$01
        ldi             mpr2,$1C
        out             OCR1AH,mpr                              ; value $11C (middle forward)
        out             OCR1AL,mpr2                             ; about 1.75ms
        ldi             mpr,$01
        ldi             mpr2,$025               ; (right tire)
        out             OCR1BH,mpr                              ; value $125 (middle forward)
        out             OCR1BL,mpr2                             ; 1.25ms
        ldi             temp0,$C3                               ;@@@@@@@@
        out             PortB,temp0                             ;@@@@@@@@
        ldi             Delay3,$10
        call    DLY
        rjmp    mainloop
turn_left:
        ldi             mpr,$01
        out             OCR1AH,mpr
        ldi             mpr,$21                                 ; value $119 (small reverse)
        out             OCR1AL,mpr                              ; 1.45ms
        ldi             mpr,$01
        out             OCR1BH,mpr
        ldi             mpr,$30                                 ; value $130 (big forward)
        out             OCR1BL,mpr                              ; 1.0ms
        ldi             temp0,$0F                               ;@@@@@@@@
        out             portB,temp0                                     ;@@@@@@@@
        ldi             Delay3,$10
        call    DLY
        rjmp    mainloop
turn_right:
        ldi             mpr,$01
        out             OCR1BH,mpr
        ldi             mpr,$20                                 ; value $122 (small reverse)
        out             OCR1BL,mpr                                  ; 1.40ms
        ldi             mpr,$01
        out             OCR1AH,mpr
        ldi             mpr,$10                                     ; value $110 (big forward)
        out             OCR1AL,mpr                                  ; 1.0ms
        ldi             temp0,$F0                                   ;@@@@@@@@
        out             portB,temp0                                         ;@@@@@@@@
        ldi             Delay3,$10
        call    DLY
        rjmp    mainloop
;**********************************************

;*SUBROUTINES AND ENDS TO INTERRUPT ROUTINES!!**
;**************Subroutines*********************

;**********LCDdelay subroutine******************
LCDdelay:                                   ; toggle PortA's pin 6 (enable pin to LCD)
        out             PORTA,temp2                         ; load portA onto temp2
        ori             temp2,0b01000000    ; force pin 6 to be high/on
        out             PORTA,temp2                         ; output new value to PortA
        andi    temp2,0b10111111    ; force pin 6 to be low/off
        out             PORTA,temp2                         ; output new value to PortA
Del:                                                ; create delay
        dec     Delay1
        brne    Del
        dec             Delay2
        brne    Del
        ret                                                         ; return from "LCDdelay" subroutine
;**********servo_delay subroutine******************
servo_delay:
        ldi             Delay3,$18
DLY:
        dec             Delay1
        brne    DLY
        dec             Delay2
        brne    DLY
        dec             Delay3
        brne    DLY
        ret
;**********dealer_card subroutine******************
dealer_card_sub1:                   ; store value from ZL into dealer_card (dealer's shown card)
        sts             dealer_card,ZL
        ldi             temp3,$0F                                   ;@@@@@
        out             PortB,temp3                                 ;@@@@@
        inc             temp0                       ; increament the card number your looking at ($00 --> $01)
        sts             card_number,temp0           ; load new value into "card_number" (value = $01)
        ldi             temp2,$87                   ; (line up and down)
        call    LCDdelay
        ldi             temp2,$8C
        call    LCDdelay
        ldi             temp2,$82                   ; (blank)
        call    LCDdelay
        ldi             temp2,$80
        call    LCDdelay
        reti                                                ; return from UART interrupt
;**********player_card_one subroutine******************
player_card_one1:                   ; store value of your first card into "players_hand_count"
        sts             players_hand_count,ZL
        ldi             temp3,$F0                                   ;@@@@@
        out             PortB,temp3                                 ;@@@@@
        inc             temp0                       ; increament the card number your looking at ($01 --> $02)
        sts             card_number,temp0   ; load new value into "card_number" (value = $02)
        reti                                        ; return from UART interrupt
;**********do_Split subroutine******************
do_Split1:                                  ; rountine that actually calls to "Split"
        ldi             mpr,$ED
        out             OCR0,mpr                    ; move 1 of split bet pattern
;                                                   (S)
        ldi             temp2,$85
        call    LCDdelay
        ldi             temp2,$83
        call    LCDdelay
;                                               (p)
        ldi             temp2,$87
        call    LCDdelay
        ldi             temp2,$80
        call    LCDdelay
;                                               (l)
        ldi             temp2,$86
        call    LCDdelay
        ldi             temp2,$8C
        call    LCDdelay
;                                               (i)
        ldi             temp2,$86
        call    LCDdelay
        ldi             temp2,$89
        call    LCDdelay
;                                               (t)
        ldi             temp2,$87
        call    LCDdelay
        ldi             temp2,$84
        call    LCDdelay
        call    servo_delay                             ; delay longer for "arm" to finish its first move
        ldi             mpr,$EF
        out             OCR0,mpr                        ; move 2 of split bet pattern
        rjmp    recheck
;**********do_Double subroutine*****************
do_double1:                                     ; rountine that actually calls to Double Down ("DD")

        ldi             mpr,$F1
        out             OCR0,mpr                        ; move 1 of Doubling Down bet pattern
;                                               (D)
        ldi             temp2,$84
        call    LCDdelay
        ldi             temp2,$84
        call    LCDdelay
;                                               (D)
        ldi             temp2,$84
        call    LCDdelay
        ldi             temp2,$84
        call    LCDdelay
        call    servo_delay                             ; delay longer for "arm" to finish its first move
        ldi             mpr,$EF
        out             OCR0,mpr                        ;  move 2 of Doubling Down bet pattern
        rjmp    done
        ;**********do_Hit subroutine*****************
do_hit1:                                ; rountine that actually calls to "Hit"
;                                               (H)
        ldi             temp2,$84
        call    LCDdelay
        ldi             temp2,$88
        call    LCDdelay
;                                               (i)
        ldi             temp2,$86
        call    LCDdelay
        ldi             temp2,$89
        call    LCDdelay
;                                               (t)
        ldi             temp2,$87
        call    LCDdelay
        ldi             temp2,$84
        call    LCDdelay
        rjmp    recheck
;**********do_Stand subroutine*****************
do_Stand1:                                      ; rountine that actually calls to "Stand"
;                                               (S)
        ldi             temp2,$85
        call    LCDdelay
        ldi             temp2,$83
        call    LCDdelay
```

```
;                                               (t)
        ldi             temp2,$87
        call    LCDdelay
        ldi             temp2,$84
        call    LCDdelay
;                                               (a)
        ldi             temp2,$86
        call    LCDdelay
        ldi             temp2,$81
        call    LCDdelay
;                                               (n)
        ldi             temp2,$86
        call    LCDdelay
        ldi             temp2,$8E
        call    LCDdelay
;                                               (d)
        ldi             temp2,$86
        call    LCDdelay
        ldi             temp2,$84
        call    LCDdelay
        rjmp    done

;*************JUMPING ZONE #1 !!!!*************
dealer_card_sub:
        rjmp    dealer_card_sub1
recheck:
        rjmp    recheck1
player_card_one:
        rjmp    player_card_one1
;*********************************************

;*********INTERRUPT SERVICE ROUTINE!!!*********
;******Scanner!!! (start of card anylasis)*****
Scan:                                   ; Card has just been read
        in              ZL,UDR                          ; reads data in and stores it in Low 8-bit Z register
        subi    ZL,0x30                         ; subtract $30 from data read (change from ASCII to card value)

        rjmp    output_number

back:
        ldi             temp2,$82                       ; (blank)
        call    LCDdelay
        ldi             temp2,$80
        call    LCDdelay
        cpi             ZL,$00                          ; compare ZL to $00, if ZL is not $00,...
        brne    not_10                          ; ...then skip next line
        ldi             ZL,$0A                          ; replace $00 with $0A in ZL register (10 value)
not_10:  lds    temp0,card_number   ; load temp0 with "card_number"
        cpi             temp0,$00                       ; is this the dealer's card?
        breq    dealer_card_sub         ; if so, then jump to "dealer_card_sub"
        cpi             ZL,$01                          ; compare ZL to $01, if ZL is not $01,...
        brne    not_Ace                         ; ...then skip next two lines
        ldi             temp3,$01
        sts             soft,temp3                      ; flag for going to the soft table later
not_Ace: cpi    temp0,$01       ; is this the player's first card?
        breq    player_card_one         ; if so, then jump to "player_card_one"
        cpi             temp0,$02                       ; is this the player's second card?
        brne    no_split_no_dd1         ; if NOT, then jump to "no_split_no_dd1"
        lds             temp0,players_hand_count ; is this card value the same as...
        cp              temp0,ZL                        ; ...the first card (i.e. current hand count)
        breq    question_split          ; is so then jump to "question_split"
;       lds             temp0,players_hand_count                ;@@@@@@?@@@????
        cpi             temp0,$01                       ; is the first card an Ace?
        breq    question_double2        ; if so, then check for doubling down
        cpi             ZL,$01                          ; is the second card an Ace?
        breq    question_double2        ; if so, then check for doubling down
        add             temp0,ZL                        ; add values and see if total equals 9, 10, 11
        cpi             temp0,$0B                       ; is it 11?
        breq    do_double                       ; if so, then ALWAYS double down!
        cpi             temp0,$0A                       ; is it 10?
        breq    question_double         ; is so then check for doubling down
        cpi             temp0,$09               ; is it 9?
        breq    question_double ; is so then check for doubling down

no_split_no_dd1:                ; NO special bets! only soft or hard...
        lds             temp3,card_number
        inc             temp3                           ; increment "card_number"
        sts             card_number,temp3

        lds             temp0,players_hand_count
        add             temp0,ZL                ; NOTE: adding must be done here (again),...
        sts             players_hand_count,temp0        ; ...incase above adding was skipped

        lds             temp0,final_card_number         ; load temp0 with number of cards player has (i.e. 2 -> 3 -> 4...)
        out             PortB,temp3                                     ; @@@@@@
        dec             temp3                   ; NOTE: temp3 = "card_number" value from above
        cp              temp0,temp3                     ; compare final card with (card_number - 1)
        breq    hard_or_soft                            ; if equal then no more players cards to read, so jump to
hard_or_soft
        reti                                                    ; return from UART interrupt
hard_or_soft:                   ; go to hard table or soft table?
        lds             temp3,soft
        cpi             temp3,$01                               ; checking if "soft flag" is set
        breq    soft_table                              ; if so then branch to "soft_table"
        rjmp    hard_table                              ; else jump to "hard_table"

;***************JUMP ZONE #2 !!!!**************
do_split:
        jmp             do_split1
do_double:
        jmp             do_double1
do_hit:
        jmp             do_hit1
do_stand:
        jmp             do_stand1
question_double2:
        jmp             question_double21
no_split_no_dd:
        jmp             no_split_no_dd1
;*********************************************

;***********BASIC PLAY TABLES/SYSTEMS*********
;                               ****Split Table***
question_split:                 ; Checking to split or not to split,... that is the question
        lds             temp0,dealer_card       ; load the dealer's card into temp0 for studying
        cpi             ZL,$00                          ; compare ZL to $00, Is it a 10, J, Q, K?
        breq    no_split_no_dd          ; if so then jump to "no_split_no_dd"
        cpi             ZL,$01                          ; compare ZL to $01, Is it a Ace?
        breq    do_split                ; if so then jump to "do_split"
        cpi             ZL,$02                          ; compare ZL to $02, Is it a 2?
        breq    do_split                                        ;+++++++++
        cpi             ZL,$03                          ; compare ZL to $03, Is it a 3?
        breq    do_split                                        ;+++++++++
        cpi             ZL,$04                          ; compare ZL to $04, Is it a 4?
        breq    do_split                                        ;+++++++
        cpi             ZL,$05                          ; compare ZL to $05, Is it a 5?
        brne    s6                                      ; if not then check for 6's
        lds             temp0,players_hand_count ; if 5, then...
        add             temp0,ZL                ; ...add temp0 and players_hand_count,...
        rjmp    question_double         ; ...then jump to question_double!
s6:     cpi             ZL,$06                          ; compare ZL to $06, Is it a 6?
        breq    do_split                                        ;+++++++++
        cpi             ZL,$07                          ; compare ZL to $07, Is it a 7?
        breq    do_split                                        ;+++++++
        cpi             ZL,$08                          ; compare ZL to $08, Is it a 8?
        breq    do_split

        cpi             ZL,$09                          ; compare ZL to $09, Is it a 9?
        breq    do_split                                        ;+++++++
        reti                                    ; NOTE: not a valied card number! (leave)
```

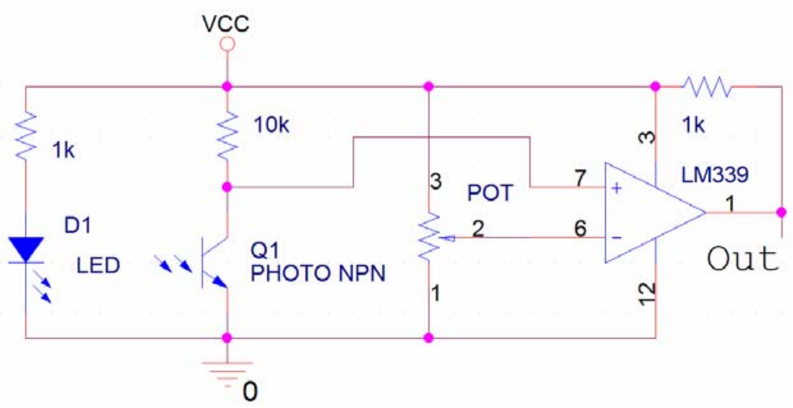
```
;                               ****Double Down Table***
; Checking to double or not to double,... that is the question
question_double:        ; player has 9 or 10 total, find out if player should double down
        lds             temp3,dealer_card       ; loads the dealers card into temp3 for choosing
        cpi             temp3,$0A               ; if dealers card is a 10
        breq    no_split_no_dd          ; then don't dd (leave)
        cpi             temp3,$0B               ; if dealer's card is an Ace
        breq    no_split_no_dd          ; then don't dd (leave)
        cpi             temp0,$0A               ; compare temp0 to $10, Is total 10?
        breq    do_double               ; if you have 10, then dd
        cpi             temp3,$09               ; if...$09
        breq    no_split_no_dd          ; then don't dd (leave)
        cpi             temp3,$08               ; if...$08
        breq    no_split_no_dd          ; then don't dd (leave)
        cpi             temp3,$08               ; if...$07
        breq    no_split_no_dd          ; then don't dd (leave)
        rjmp    do_double               ; dealers card must be less then 7, player must have 9 (DD)
question_double21:      ;****player has an A in first two cards, should he/she double?****
        lds             temp3,dealer_card       ; loads the dealer's card into temp3 for studying
        rjmp    do_double                       ;++++++++

;                               ****Soft table (no DD option)***
soft_table:
        inc             temp0
        sts             final_card_number,temp0
        lds             temp0,players_hand_count
        ldi             temp0,$55
        out             PortB,temp0
        rjmp    do_stand                        ;++++++

;                               ****Hard table (no DD option)***
hard_table:
        inc             temp0
        sts             final_card_number,temp0
        lds             temp0,players_hand_count
        cpi             temp0,$10
        breq    h16
        andi    temp0,0b11110000
        cpi             temp0,0b00010000
        breq    d_stand                         ;++++++
h16:
        rjmp    do_hit                          ;++++++

;**************JUMP ZONE #2 !!!!**************
d_stand:
        rjmp            do_stand1
;*********************************************

;************CLOSING HAND!*******************
recheck1:                                       ; end of your choice (Hit/Split)
        clr             temp0
        sts             card_number,temp0
        sts             dealer_card,temp0
        sts             soft,temp0
        sts             players_hand_count,temp0
;       ldi             temp3,$18                                       ;@@@@@
;       out             PortB,temp3                                     ;@@@@@
        reti                                            ; leave interrupt service routine

done:                                                   ; end of your hand (DD/Stand)
        ldi             temp0,$02
        sts             final_card_number,temp0
        clr             temp0
        sts             card_number,temp0
        sts             dealer_card,temp0
        sts             soft,temp0
        sts             players_hand_count,temp0
;       ldi             temp0,$99                                       ;@@@@@@@@
;       out             PORTB,temp0                                     ;@@@@@@@@
```

```
            reti                                                                    ; leave interrupt service routine

output_number:
            lds                     temp0,card_number       ; load temp0 with "card_number"
            cpi                     temp0,$00                           ; is this the dealer's card?
            brne        no_clear
            ldi                     temp2,$00               ; (Clear Home)
            call        LCDdelay
            ldi                     temp2,$01
            call        LCDdelay
no_clear:
            cpi                     ZL,0x00                             ; compare ZL to $00, if
            breq        zero                                ; ZL(data - $30) is $00 then jump to zero
            cpi                     ZL,0x01
            breq        one
            cpi                     ZL,0x02
            breq        two
            cpi                     ZL,0x03
            breq        three
            cpi                     ZL,0x04
            breq        four
            cpi                     ZL,0x05
            breq        five
            cpi                     ZL,0x06
            breq        six
            cpi                     ZL,0x07
            breq        seven7
            cpi                     ZL,0x08
            breq        eight8
            cpi                     ZL,0x09
            breq        nine9
            reti                    ;(note not a valied card number!) ; leave interrupt service routine
seven7:
            jmp                     seven
eight8:
            jmp                     eight
nine9:
            jmp                     nine
zero:
            ldi                     temp2,$83                           ; (10)
            call        LCDdelay
            ldi                     temp2,$81
            call        LCDdelay
            ldi                     temp2,$83
            call        LCDdelay
            ldi                     temp2,$80
            call        LCDdelay
            rjmp        back                                                        ; leave interrupt service routine
one:
            ldi                     temp2,$84
            call        LCDdelay
            ldi                     temp2,$81
            call        LCDdelay
            rjmp        back                                                        ; leave interrupt service routine
two:
            ldi                     temp2,$83
            call        LCDdelay
            ldi                     temp2,$82
            call        LCDdelay
            rjmp        back                                                        ; leave interrupt service routine
three:
            ldi                     temp2,$83
            call        LCDdelay
            ldi                     temp2,$83
            call        LCDdelay
            rjmp        back                                                        ; leave interrupt service routine
four:
            ldi                     temp2,$83
            call        LCDdelay
            ldi                     temp2,$84
```

```
        call    LCDdelay
        rjmp    back                                    ; leave interrupt service routine
five:
        ldi             temp2,$83
        call    LCDdelay
        ldi             temp2,$85
        call    LCDdelay
        rjmp    back                                    ; leave interrupt service routine
six:
        ldi             temp2,$83
        call    LCDdelay
        ldi             temp2,$86
        call    LCDdelay
        rjmp    back                                    ; leave interrupt service routine
seven:
        ldi             temp2,$83
        call    LCDdelay
        ldi             temp2,$87
        call    LCDdelay
        rjmp    back                                    ; leave interrupt service routine
eight:
        ldi             temp2,$83
        call    LCDdelay
        ldi             temp2,$88
        call    LCDdelay
        rjmp    back                                    ; leave interrupt service routine
nine:
        ldi             temp2,$83
        call    LCDdelay
        ldi             temp2,$89
        call    LCDdelay
        rjmp    back                                            ; leave interrupt service routine

;*******************************************************************************
;**************Variables Defined!!***************************************
.org    $500
.dseg
card_number:            .byte 1         ; The number of the player's card being looked at/next,
                                                ; Note: the dealer's card is "card_number" zero ( $00 )
dealer_card:            .byte 1         ; value of dealer's card
players_hand_count:     .byte 1         ; players total hand count
final_card_number:      .byte 1         ; holds the value that card_number is counting up to!
soft:                   .byte 1          ; pin-0 is used as a flag for when player's hand is soft
```

Line tracking ciruit:

(from William Dubel’s report on line tracking)



Circuit outputs logic high when a black line is detected.