Authors: Damian Bąkowsk, Iga Bubula

Supervisor: Sebastian Koryciak

# Microprocessors Project Report

Black Jack game on ATmega 328PB in C language

## Introduction:

Classic Blackjack rules:

- Game uses standard 52 card deck
- An ace is usually equivalent to 1 or 11 points, whilst 10s/Jacks/ Queens /Kings count as ten points and cards marked 2 to 9 are valued according to their pip value.
- Player attempts to beat the dealer by getting a count as close to 21 as possible without going over 21.
- The value of a hand is equivalent to the sum of the total cards.
- A ‘blackjack’ is considered to be the highest hand, which in essence normally consists of two cards; an ace and a king/ queen/ jack, which would be equivalent to the number 21. The second highest hand would be a ‘21’ which consists of 3 to 5 cards which are of the value of 2-9.
- During the initial phase of the game, the dealer deals two cards to the player as well as himself. Then he will display one of his cards for the player to see whilst the other one, which is known as the ‘hole card’, will be displayed face down.
- The player after receiving his two cards can do one of two actions: take or stand.
- If player decide to stand then dealer will start their own turn. They can also decide to take card or stand.

Our modifications/simplifications due to implementation process:

- An ace is always equivalent to 11 points.
- Suits are not distinguishable.
- No bets.
- Dealer strategy: If the total is 16 or more, it must stand. In other cases, it must take a card.

## Technical Documentation:

Port description and used peripherals:

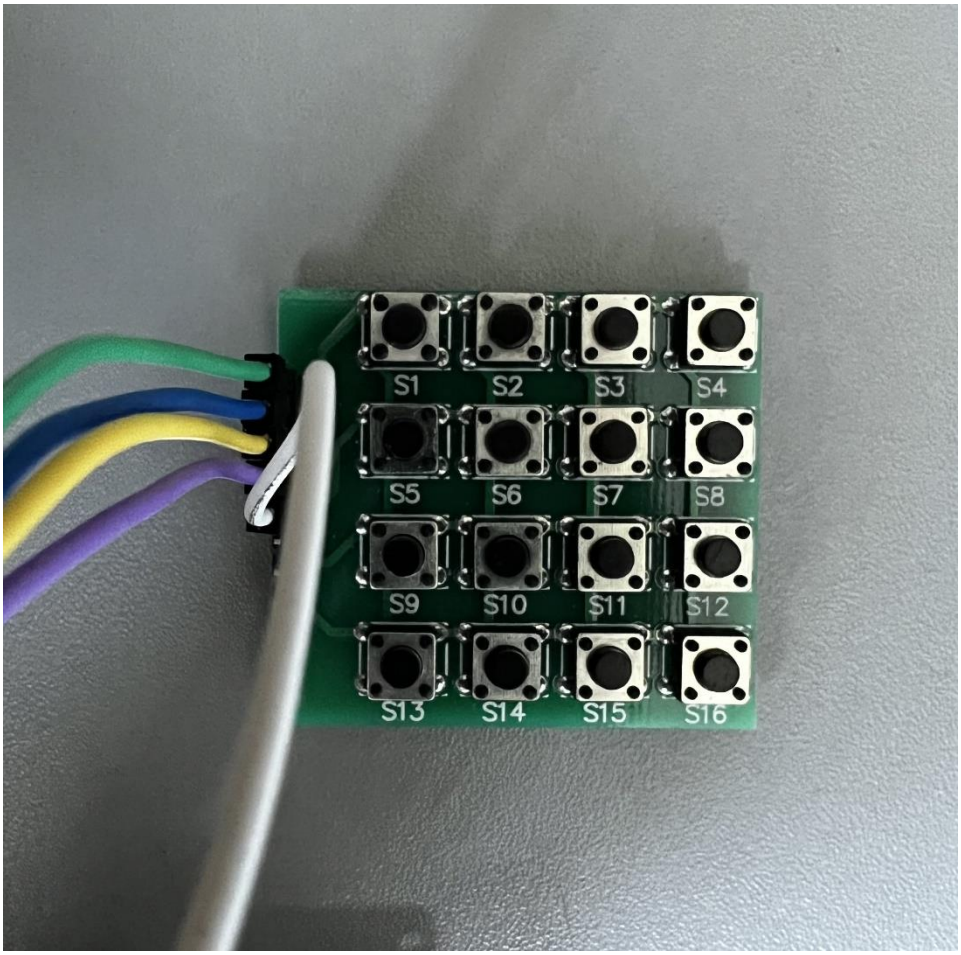



*Fig. 1: PORTB: buttons (PCINT0) interrupt is connected to that specified port*

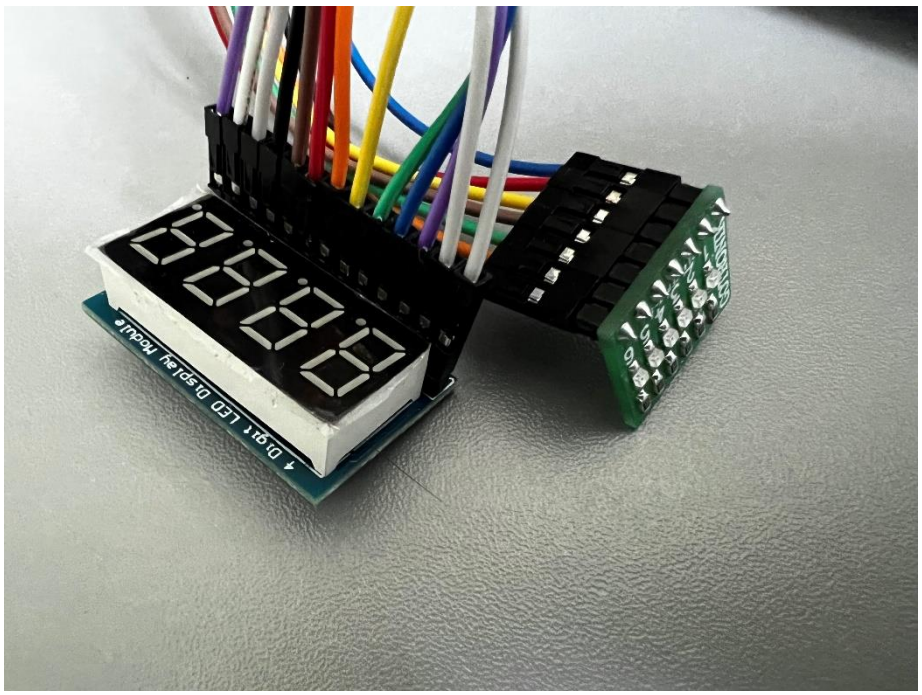


*Fig. 2: PORTC: LEDs*

*PORTD: 7 Segment Display (A:G) – value to display*

*PORTE: 7 Segment Display (D1:D4) – selecting display*

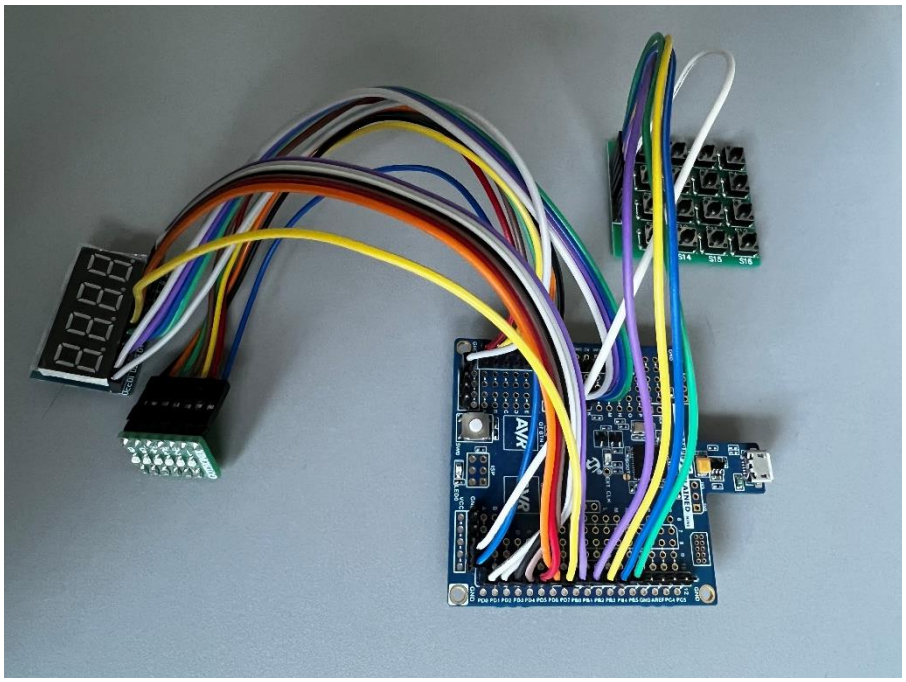
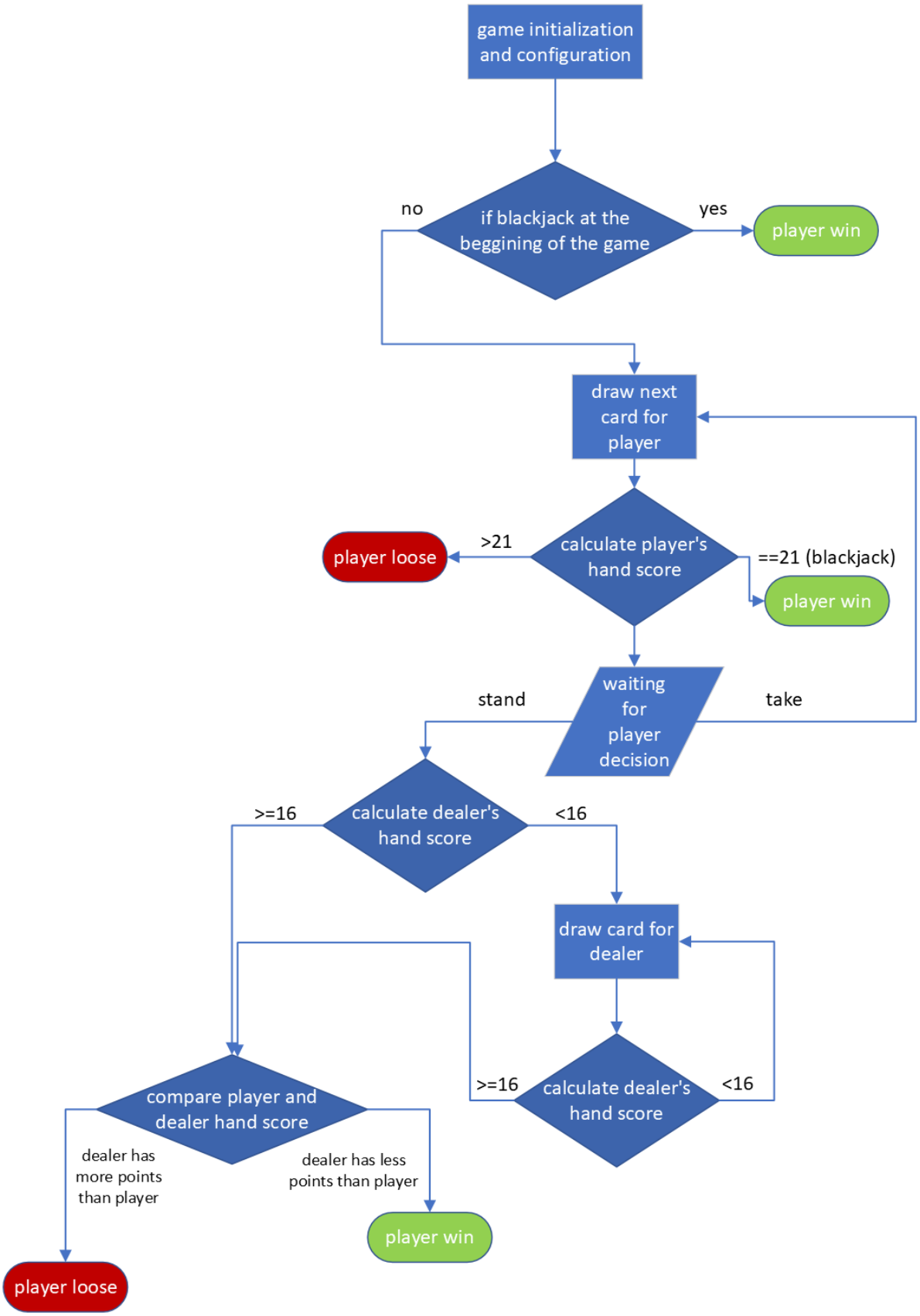


*Fig. 3: Whole setup*

Block diagram:



Global constants:

`const unsigned char deck[]` – consist 12 cards without distinguishing suits

`const unsigned int digitsHex[]` – consists numerical values in hexadecimal form for displaying on screen purposes from 1 to 9

`const unsigned int facesHex[]` – consists letters ‘J’ ‘Q’ ‘K’ ‘A’ for displaying on screen purposes

`const unsigned int winMessageHex[]` – consists letters used for displaying message for winner on a screen

`const unsigned int looserMessageHex[]` – consists letters used for displaying message for looser on a screen

Global variables:

`volatile unsigned char card` – for storing drawed card
`volatile unsigned char playerHand[2]` – array for storing player’s cards
`volatile unsigned char dealerHand[2]` - array for storing dealer’s cards
`volatile unsigned int playerScore=0` - variable for storing player score
`volatile unsigned int dealerScore=0` – variable for storing dealer score
`volatile uint8_t state = 0 –` flag for interrupt which detects which of two button was pressed (0 or 1)
`volatile uint8_t lis = 0` – flag for activating interrupt (used to secure interrupt to be called more then one time)
`volatile uint8_t oldPINB = 0xFF` – flag for detecting rising edge

Functions for card drawing:

`uint8_t trueRandom()` - returns a random number using thermal noise from the ADC.

`int cardTwoPoint(char card)` - takes card symbol and returns it’s point value

`char drawCard()` - returns drawed card based on random value from `trueRandom()`

Functions for displaying content on 7 segment display:

`int getFirstDigit(int val)` – takes two digit number and returns it first digit

`int getSecondDigit(int val)` – takes two digit number and returns it second digit

`void dispNum(int val, int period)` - displays two digit value on a 7 segment display for a selected period of time.

`void dispChars(char c, char a, char b, int period)` - displays two chars on a 7 segment display for a selected period of time.

`void dispMessage(int option)` - displays a selected dynamically moving message based on a defined global array. Depending on option given as an argument it uses another set of chars. Option == 1 displays message for winner, option == 0 displays message for looser.

Game logic and loops:

`void playerWin(void)` – calls `dispMessage(1)` function to display the message specified in global array when the player wins the game.

`void playerLoose(void)` – calls `dispMessage(0)` function to display the message specified in global array when the player loses the game.

`void playerLoop(void)` – main game loop which listen interrupts and responses on player decisions

`void dealerLogic(void)` – manages dealer turns based on defined strategy

`void finalCondition(void)` – compares player and dealer scores when there's no winner after previous steps.

`int blackjackCondition(void)` - this function checks if the blackjack occurred.

Interrupt:

`ISR(PCINT0_vect)` - interrupt for two buttons on PORTB used to detect which button was pressed (1 or 2). Due to the fact that interrupt …… we was forced to use but we made workaround to detect only rising edge manually (`oldPINB`) and avoid queued interrupt activations hence only one detection can happen then (`lis`) flag immediately is setting to 1.

`int listen(void)` – this function activate interrupt by setting global variable responsible for it, then it make blinking effects using connected diodes to show that game is waiting for player decision

Main code which calls all that above:

`void configuration(void)` - this function configures all the ports and interrupts as well as globally activates them.

`void gameInit(void)` – prepares game, draws first two cards for player and one for a dealer then displays it in right manner on display

`int main(void)` - calls all above functions in right order

## Conclusions:

Two bugs noticed so far:

- Sometimes game draws zero card which has 0 points
- Sometimes repeatability of drawed sets at the beginning of the game can be noticed.

## Appendix:

Screenshots:

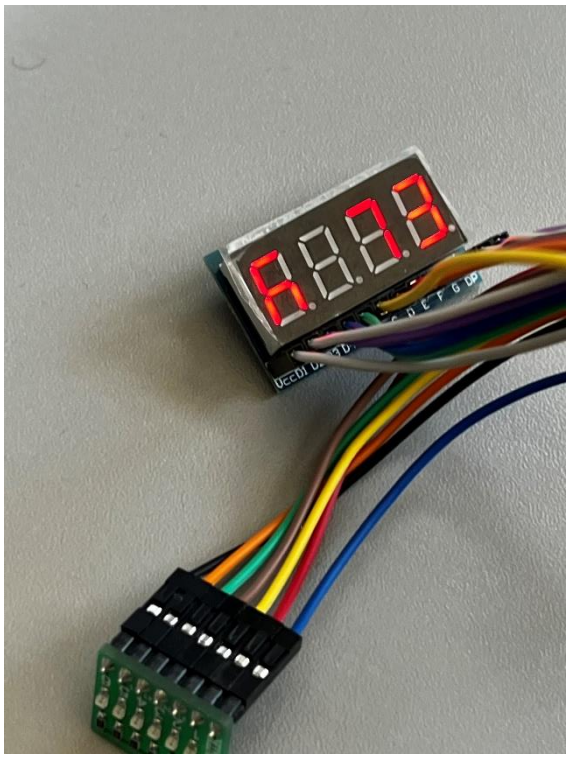


*Fig. 4: Example of dealer and player hands at the beginning of the game*

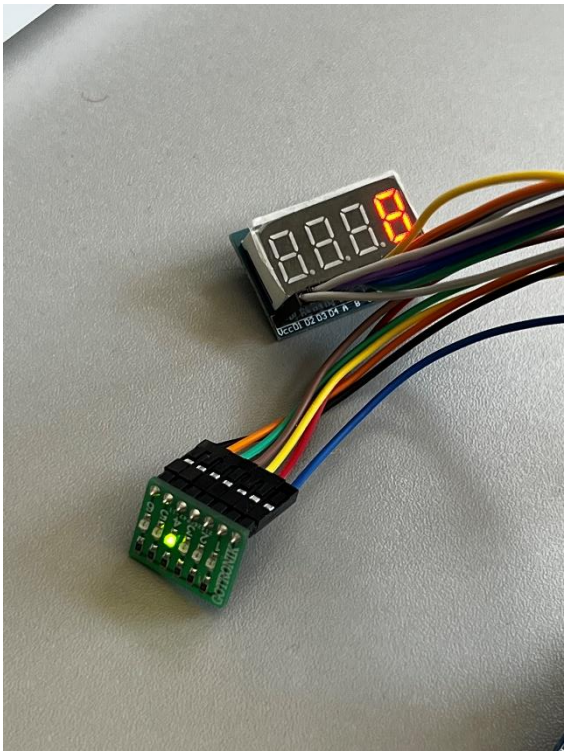


*Fig. 5: Example score and diode waiting for player response*

Movie of example gameplay:

https://aghedupl-my.sharepoint.com/:v:/g/personal/bakowski_student_agh_edu_pl/EYUcgi6JCMVGiFT-clruIVIBxOYlEdgY2lpZAubNdS0ZrA?e=9feLhh

Source code: main.c