Implementation of the Challenge of Mekon with an AVR Microcontroller

Aaron Fainman
Student number: 1386259

The Challenge of Mekon is a popular wire loop game that can be implemented on an Arduino Uno with an AVR microcontroller (MCU). Using the built-in timer, external interrupts to start and end the game, and a pin change interrupt for when the ring and wire come into contact, a very functional game can be built.

An integral element of the game is the game score. It can be increased by the timer or when the ring and wire loop come into contact. The score is stored as a decimal number in two registers: one for the units value and one for the tens value. The score could be stored as a binary number in a single register. However, keeping the score in two registers simplifies outputting the score as a double digit decimal number to a seven segment display. When both the timer interrupt and the ring-wire interrupt executes, the units score register is increased. This is then compared to a value of 10, and if equal or greater, 10 is subtracted from the units score and the tens score is incremented by 1.Thus the score remains in decimal. If true, this comparison statement also turns on an orange LED[1], since the score has jumped over a multiple of 10. The value of the penalty for the ring and wire touching is stored as a global variable, PENALTY. This allows for easy changing of its magnitude.

Gap sensors are used to begin and end the game. They are located at each end of the wire loop, and connect to schmitt triggers (SN74LS14) [1] which increase the noise margin and switching speed of the signal [1]. Thereafter, the schmitt triggers connect to their respective pins on the MCU, the INT0 and INT1 pins. This allows for the rising edge of each gap sensor signal to cause an interrupt to execute. In the first interrupt handler, the timer count is cleared, the timer interrupt is enabled, and the wire-ring interrupt is enabled. Collectively, this results in the 7 segment display displaying the score, and the score increasing both with time and when the ring and wire touch. The speaker also begins producing a sound. Thus the game has started. At the other end of the wire loop, the second gap sensor ends the game. When this interrupt handler (INT1) is called the timer interrupt is disabled, the speaker PWM is disabled and the wire-ring interrupt is disabled. Collectively, this ends the game by stopping the score from changing and the speaker from producing sound. The game can also end when the score reaches 99. Branch statements in the timer and penalty interrupt handlers are used for this and can call the same function as the interrupt handler for the second gap sensor.

When the ring and wire loop come into contact, the score will change. The wire loop is constructed from a copper pipe and is left disconnected on one side. The other side is connected to the non-inverting input of an op amp comparator (LM386) [2]. The inverting input connects to a voltage divider, and sees 3V. When the ring, which is connected to the power rail, is touching the loop, the comparator outputs 5V (logic HIGH). When the ring and loop are not in contact, the comparator outputs 0V (logic LOW). The comparator output then feeds into the input of a schmitt trigger. Finally, the output of the schmitt trigger connects to Pin D4 (PD4) of the AVR. This has been set up as a pin change interrupt, PCINT20. The ultimate effect of this is that the pin will see a HIGH signal generally. When the ring touches the loop, the pin senses a change in voltage to a LOW signal and an interrupt is called. Within this interrupt handler a red LED turns on and, the score increases, as discussed above. Additionally, the pin change interrupt is temporarily disabled. This is done since the pin change interrupt only senses a change in voltage. The disabling avoids double penalisation when a user touches the ring to the wire and then takes it off. The timer interrupt re-enables the pin change interrupt.

A 16-bit timer (timer 1) has been used for both restoring some default game settings and score incrementing. The timer interrupt executes every quarter second. When this happens, it is ensured that both the orange and red LEDs are off. Also, the pin change interrupt is enabled. Therefore, LEDs do not stay on for long and the ring-wire penalty is only disabled temporarily. Every few seconds, the score is

[1] The orange LED has been implemented using resistors. This could have been done using PWM instead, as described later in the report. However, using PWM was not as effective a solution. The PWM port for another timer is in the middle of Port B, which is already being used for outputting the score as a 4 bit binary coded decimal. Since the Arduino Uno only has one full 8 bit port (port D) which needs to be used for external interrupts, in order to do a separate score, the score would have had to be split between different ports - far more complex than simply using resistors.

also meant to increase. By comparing the number of times the timer statement has executed and a register (NUM_SECONDS), the score will increase every few seconds. An additional timer could have been used to implement these different functions. However, having one timer is more efficient than setting up two different timers. It also allows for the user to change how long it takes before the timer increments the score. Analogue-to-digital conversion (ADC) has been used to set the value in NUM_SECONDS. A potentiometer acts as a variable voltage divider. This feeds into an analogue pin of the microcontroller, set up for ADC. Only the 3 most significant bits are read, allowing for 7 different speeds at which the timer executes. This creates the effect of different game modes, ranging from very easy (4 seconds for every point) to very hard (0.5 seconds), and can be controlled by the user. During game play, however, the ADC is disabled since the time for a point should remain constant.

Another function of the timer and wire-ring interrupts is to change the pitch of the speaker. The speaker is connected to Port D5 of the MCU. Phase correct PWM has been set up in timer 0 with a compare match starting at 120. This corresponds to a frequency of 65 Hz ($f = f_{clk}/(2 \cdot N \cdot TOP)$) [3]. As the game progresses, both timer 1 and the wire-ring interrupt decreases this compare match when executed. A score of 99 corresponds, at most, to a frequency of 372 Hz. These two values fall well within the range of human hearing, from 20 Hz to 20 000 Hz [4]. As the game progresses, the increasing pitch adds to the urgency a player feels to complete the game.

To summarise the flow of the program, as seen in Figure 1, when the Arduino is connected to power, a series of setup statements occur. These involve setting up the timer and interrupt registers, as well as the ADC. When the ADC conversion completes, the data is read into a register which is used to control how many times the timer interrupt executes before a point is scored. When the game is begun, by passing the ring through a gap sensor, the pin change and the timer interrupts are enabled. Thus the score can change. This happens every few seconds and when the ring and wire loop come into contact. Both of these conditions call their own interrupt statements where the score is increased and if necessary, red or orange LEDs are turned on. The game ends when either the score reaches 99, or when the ring passes through the second gap sensor. The circuit schematic for this implementation can be seen in Figure 2. Thus with minimal components and relatively simple code, an entertaining Challenge of Mekon game can be created.

References:

[1] "SNx414 and SNx4LS14 Hex Schmitt-Trigger Inverters", *Texas Instruments*, 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/sn74ls14.pdf. [Accessed: 18- Sep- 2018].

[2] "LM386 Low Voltage Audio Power Amplifier", *Texas Instruments*, 2017. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm386.pdf. [Accessed: 17- Sep- 2018].

[3] *Atmel*, 2016. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. [Accessed: 17- Sep- 2018].

[4] R. Serway, J. Jewett and V. Peroomian, *Physics for scientists and engineers*, 9th ed. Thomson Brooks/Cole, 2004, p. 516.

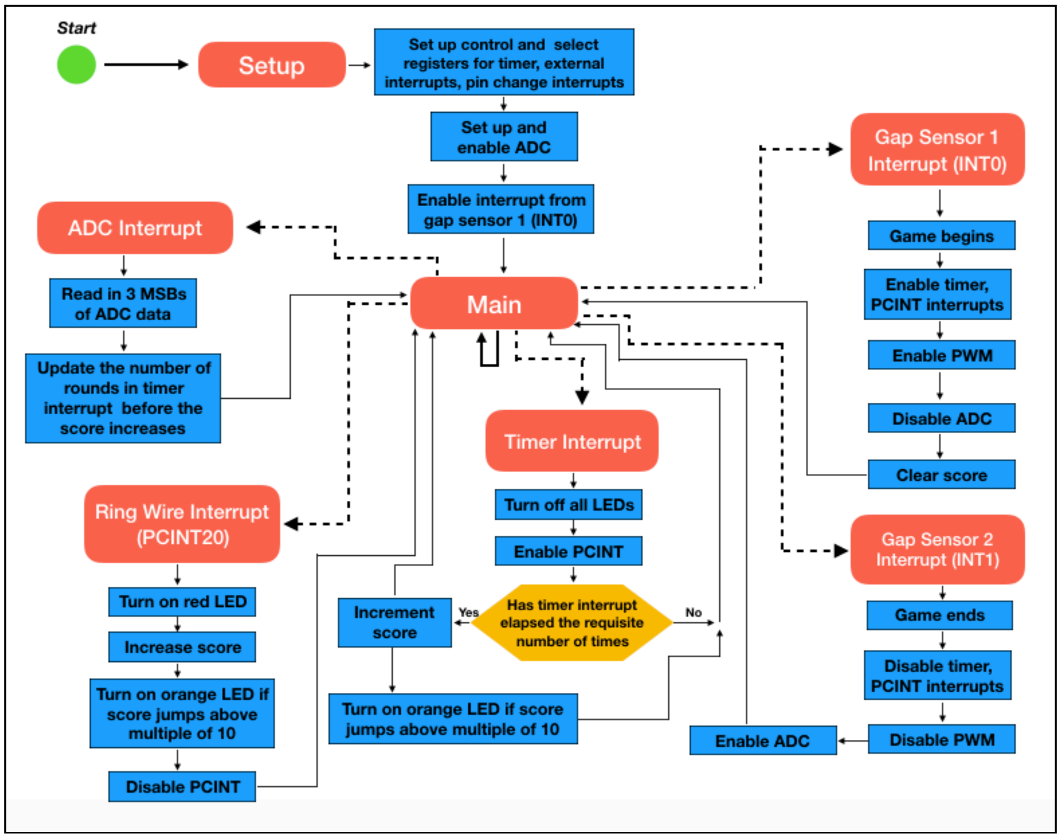
Start
Setup
Set up control and select registers for timer, external interrupts, pin change interrupts
Set up and enable ADC
Enable interrupt from gap sensor 1 (INT0)
Main
ADC Interrupt
Read in 3 MSBs of ADC data
Update the number of rounds in timer interrupt before the score increases
Gap Sensor 1 Interrupt (INT0)
Game begins
Enable timer, PCINT interrupts
Enable PWM
Disable ADC
Clear score
Timer Interrupt
Turn off all LEDs
Enable PCINT
Has timer interrupt elapsed the requisite number of times
Yes
No
Increment score
Turn on orange LED if score jumps above multiple of 10
Ring Wire Interrupt (PCINT20)
Turn on red LED
Increase score
Turn on orange LED if score jumps above multiple of 10
Disable PCINT
Gap Sensor 2 Interrupt (INT1)
Game ends
Disable timer, PCINT interrupts
Disable PWM
Enable ADC

Figure 1. Block Diagram summarising the flow of the program

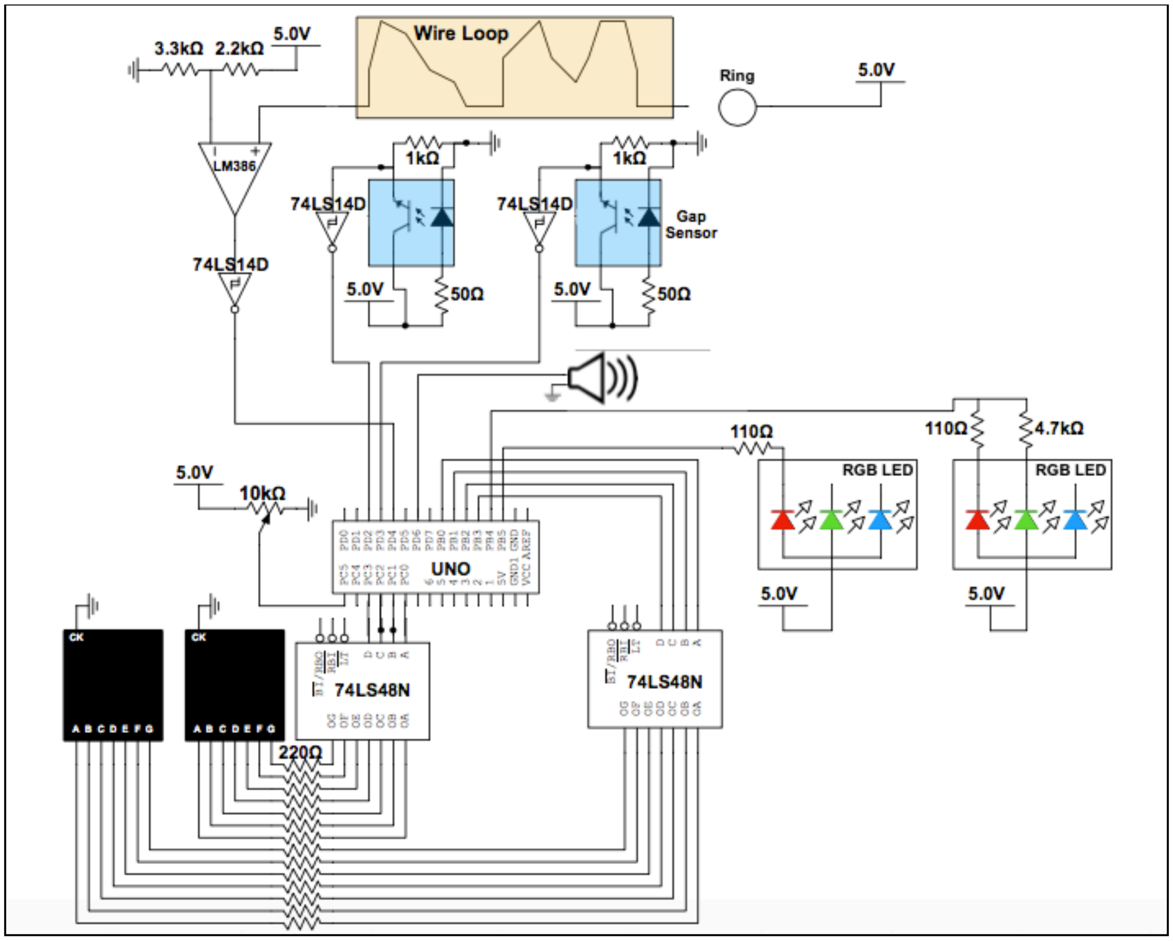

Figure 2. Full circuit schematic for the Challenge of Mekon