CSEE 4840 “WhAck-A-MoLe”

COLUMBIA UNIVERSITY

Georgios Charitos
gc2662

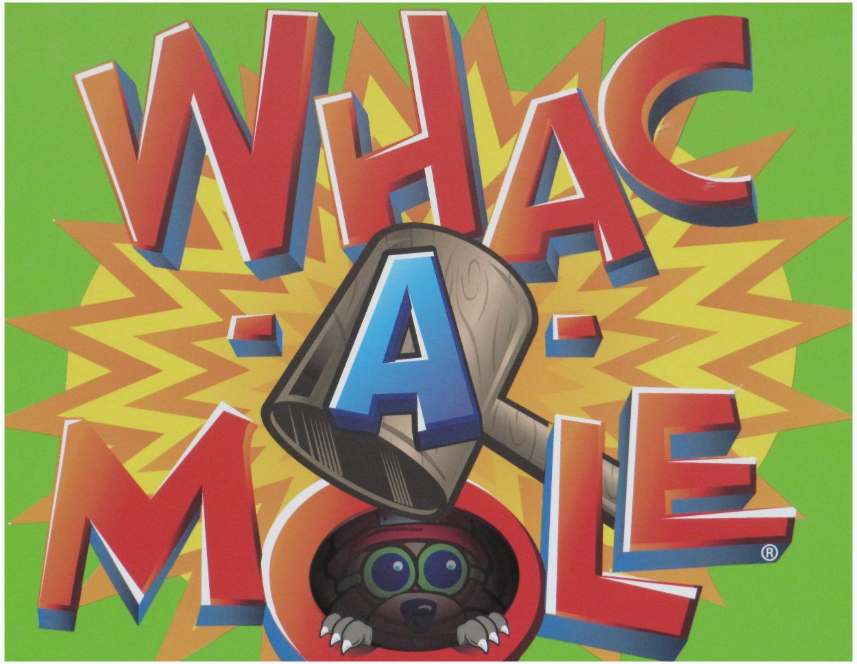
Aditya Bagri
aab2234

Jai Sharma
js4473

Astha Agrawal
aa3755

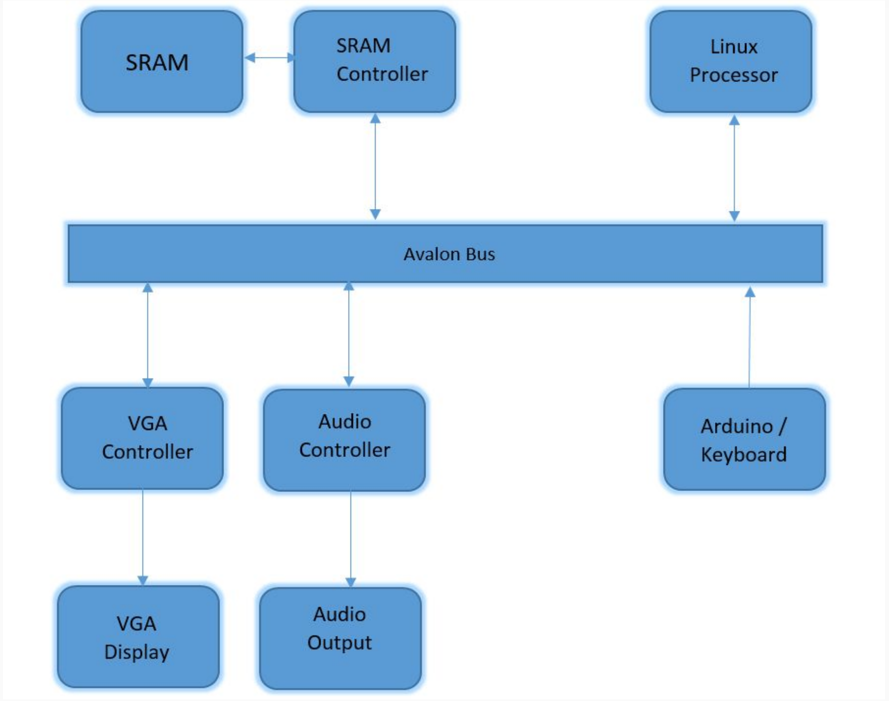
“Whac-A-Mole” The Game

- The logic of the game is straightforward: Hit as many moles as possible with the hammer.
- The time is restricted to 60 seconds per level.
- Each successful Mole hit increases the score by one.
- The player with the highest score at the end is the winner.

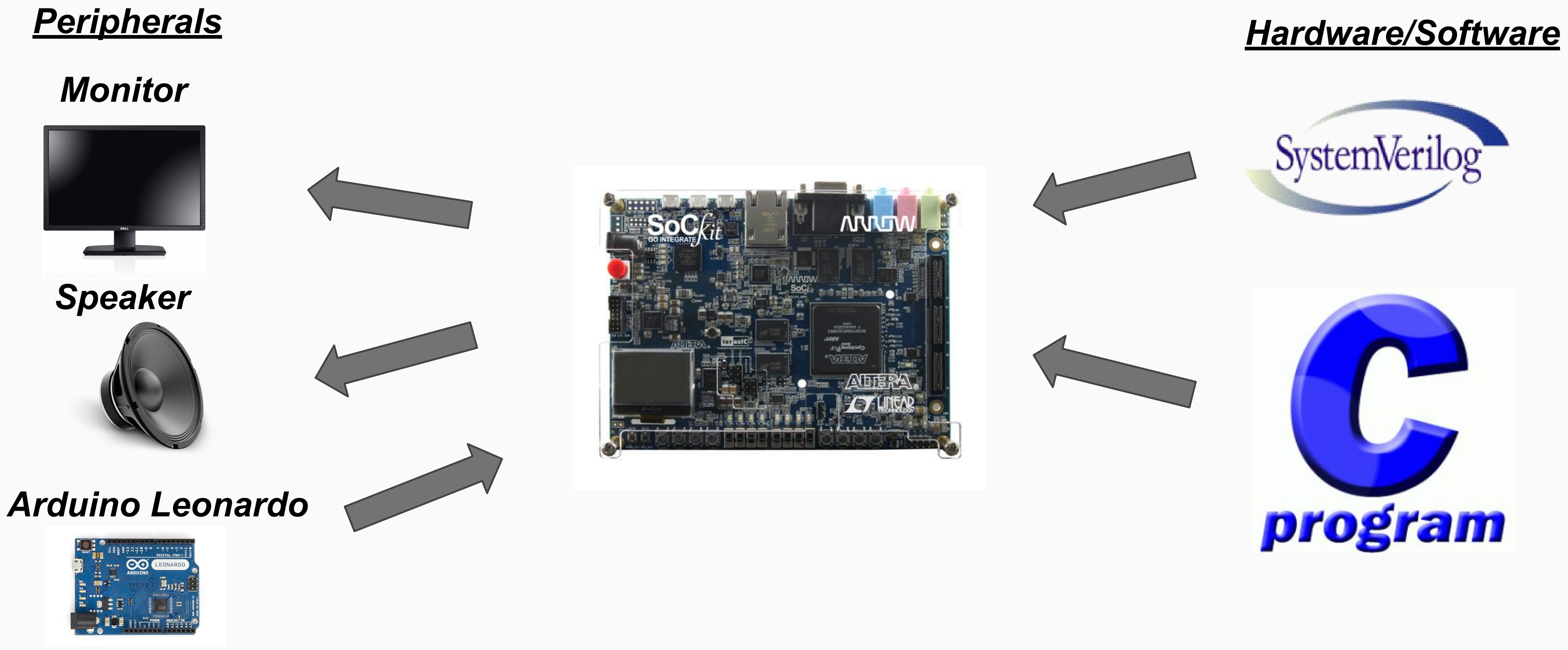


Project Architecture

The main components of the architecture for this project involve the SRAM Memory on the board, the linux processor on the computer, the Avalon bus for the Altera FPGA communication, the VGA component for the scree, Audio component and the Arduino/Keyboard for the buttons. The link between these components is shown in the image on the right and an explanation is provided in the following slides.



System Overview



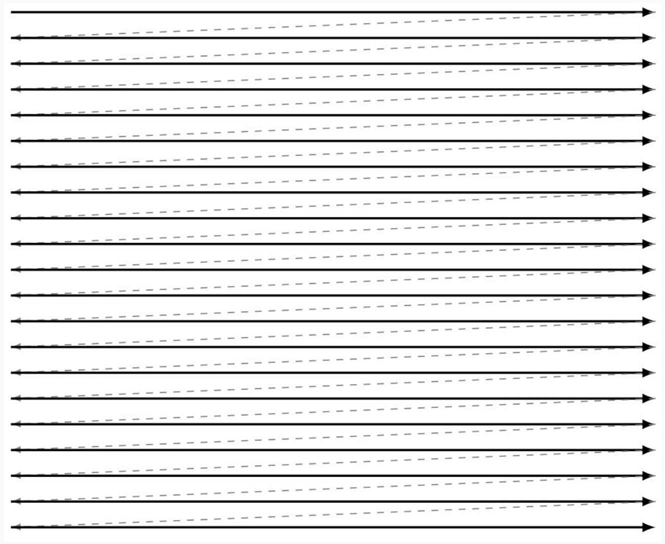
Hardware(1)

Basis of the Hardware is the raster producing 60 frames/second!!!

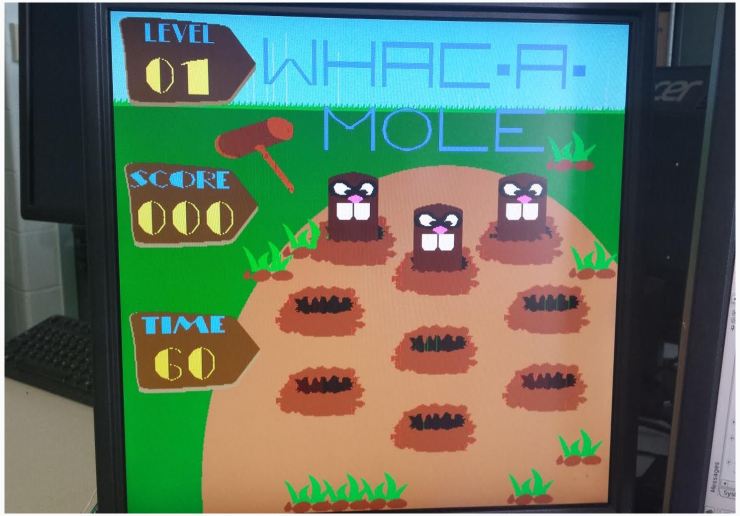
1. We used VGA_LED and VGA_LED_Emulator to produce the graphics on the monitor.
2. Background is produced in order of appearance (sky first, grass second etc).
3. Holes are ellipses that do not require Sprites.
4. Tall grass, letters, numbers, moles, hammer and mud surrounding the holes are produced with Sprites.
5. Decoders are used for producing the different numbers for (level, time, score) and different mole positions.
6. Mole popping is done using the raster(60 frames/second)

Hardware(2)

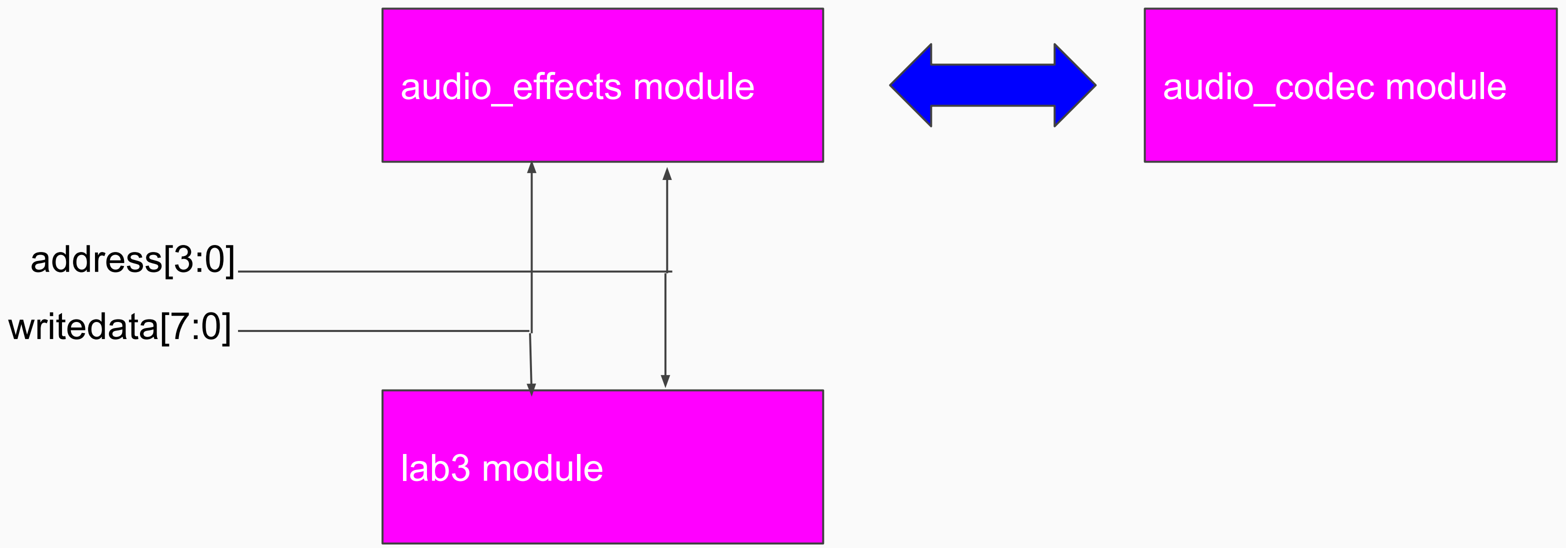
Raster Scanning



WhAck-A-MoLe Video



Hardware(3)


audio_effects module
audio_codec module
address[3:0]
writedata[7:0]
lab3 module

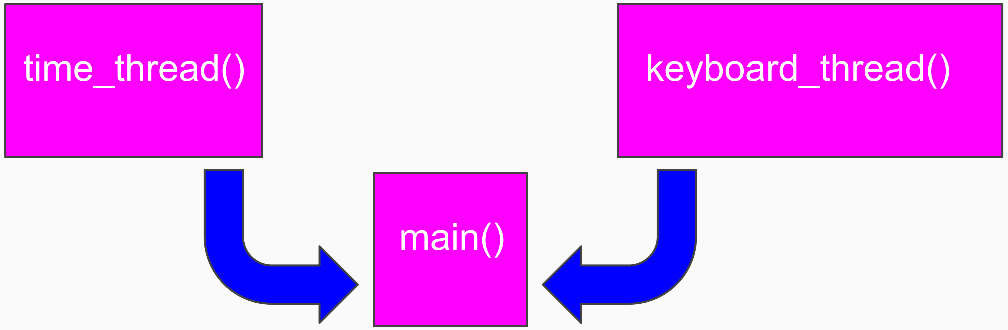
Software(1)

Main software program
hello.c

VGA_LED Device Driver
vga_led.c

Hardware
VGA_LED.sv

Main software program
hello.c



Software(2)

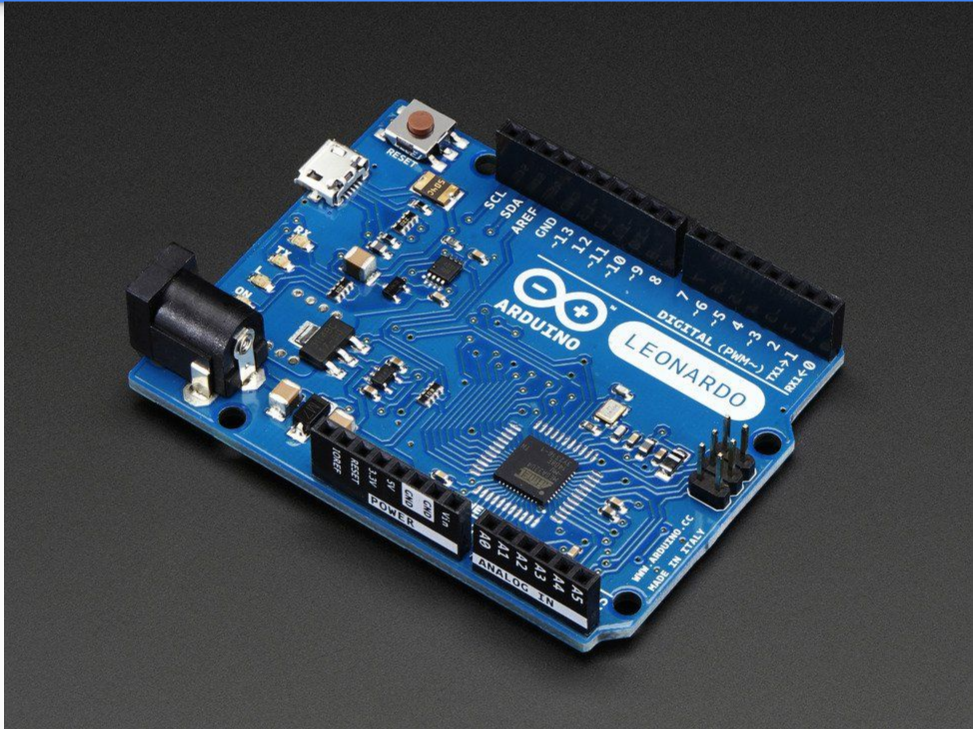
1. keyboard_thread: Arduino-Keyboard input
2. time_thread: Timer to synchronize all operations
3. main: Changes mole positions periodically via keyboard_thread, checks for mole strikes via inputs from keyboard_thread

Hardware-Software Interface

hex0[7:0]	7	6	5	4	3	2	1	0
Usage:	Mole 2 Position				Mole 1 Position			
hex1[7:0]	7	6	5	4	3	2	1	0
Usage:	Hammer Position				Mole 3 Position			
hex2[7:0]	7	6	5	4	3	2	1	0
Usage:	Sound 8	Sound 7	Sound 6	Sound 5	Sound 4	Sound 3	Sound 2	Sound 1
hex3[7:0]	7	6	5	4	3	2	1	0
Usage:	SCORE							
hex4[7:0]	7	6	5	4	3	2	1	0
Usage:	TIME							
hex5[7:0]	7	6	5	4	3	2	1	0
Usage:	LEVEL							

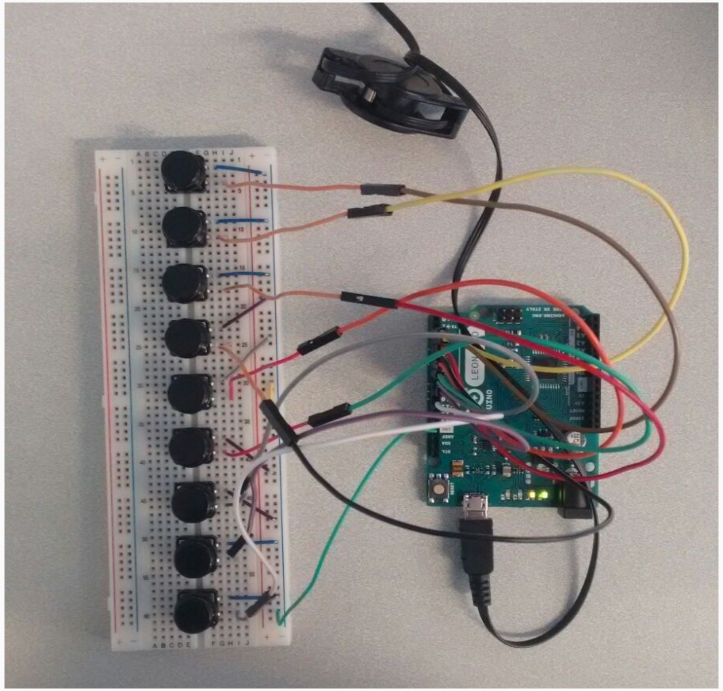
Arduino Leonardo

The Arduino Leonardo is a microcontroller board based on the exciting USB-enabled ATmega32u4 (datasheet). It contains everything needed to support the microcontroller; one simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



Why do we need the Arduino Leonardo?

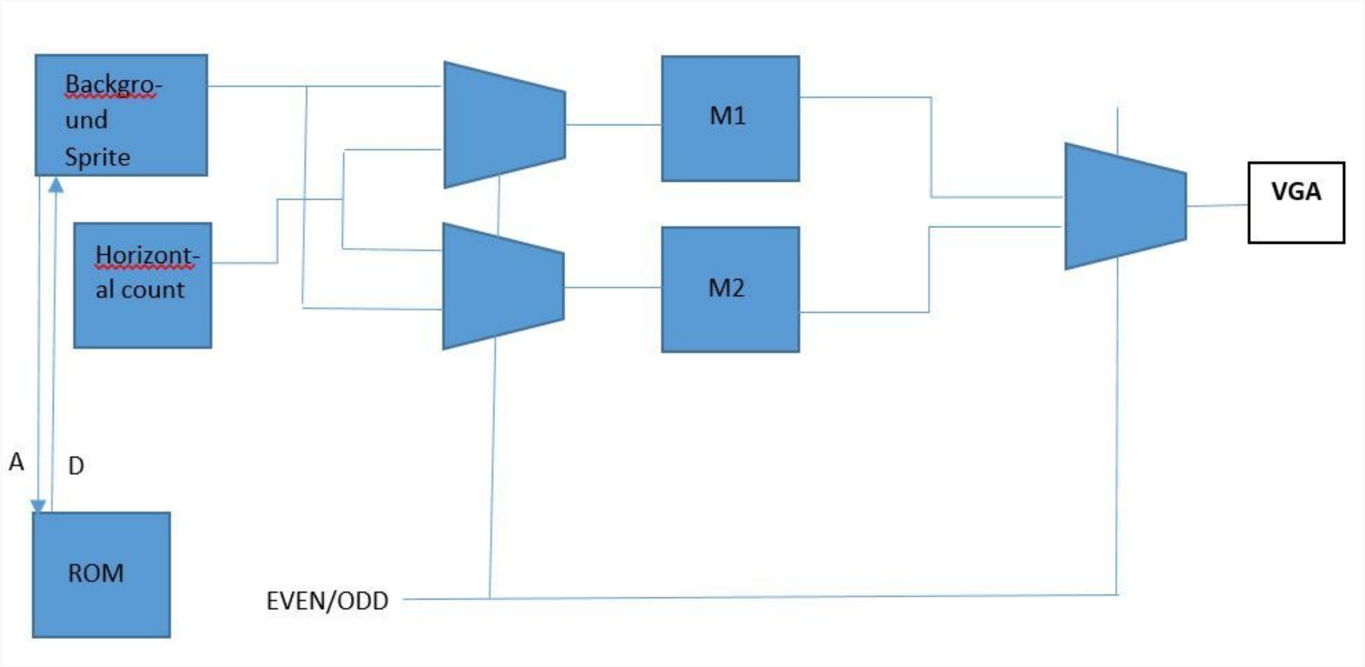
- The purpose of the game is to hit the popping mole on the screen.
- While this function can be achieved using the keys on the keyboard, it is far more elegant to have real push buttons for this task.
- The Arduino Leonardo allows us to interface push buttons on a breadboard with the Cyclone V board.
- The Arduino can be programmed to behave as a keyboard based on the button pressing.
- Eventually, we can have a push button system hooked up to the Cyclone V board via the Arduino and push the appropriate buttons to Whac the corresponding Mole.



Sprite Graphics

- The architecture for the Sprite graphics involves the RAM and a Mux for displaying the shapes on the screen.
- We can access the RAM using two dimensional arrays in SystemVerilog.
- The Mux is used for selecting the position of the particular Sprite on the screen.
- The logic we used to create the Sprites involves a combination of 1’s and 0’s as elements of the array.
- First we select the right dimensions of the array based on the size of the object we wish to see on the screen.
- Then we fill out the 2D array in the shape of the object by putting a 1 at all positions that we want to see the object and 0 everywhere else.
- And finally, we design the Mux (VGA_RGB) and Decoder for the proper positioning and logic of the appearance of the Sprites.
- For complex figures, we need to break down the memory access as multiple overlapping Sprites for a single object.

Sprite Graphics



Incomplete Goals & Lessons Learned

Incomplete Goals

- Arduino Leonardo
- Control Soundwaves

Lessons Learned

- It is better to split the design in small modules.
- Approach issues in parallel. Not in series. The ones in the end might require more time than planned. By dealing with them for a certain amount of time in the beginning you do more accurate predictions about the workload.

Future Work

1. Insert software logic for increasing mole speed for every next level.
2. Add more sophisticated graphics (mole strike, wind moving, grass, birds, cows in the background)
3. Add more mole stripes.
4. Incorporate main menu and player name interface.
5. Control soundwaves.

Conclusion

We are thrilled to present a working demonstration of the Whac-A-Mole game. The final version of the game allows the user to Whac the Moles using the keyboard buttons and records the score simultaneously for successful hits. The time counts down from 60 seconds for each level and the final score is displayed on the screen!

Acknowledgement

- We would like to take this opportunity to thank Professor Stephen Edwards for such an interesting and intellectually stimulating Embedded Systems course, we sure learned a great deal from it about hardware and software. Also, Prof. Edwards was constantly available for advice, guidance and assistance during the course of this project and we are extremely grateful for that.
- We would also like to thank the three TA's for this course for all their help during the assignments as well as the project work.