LOG BOOK RECORD: SESSION #2

Date & Time: 1st April 2018 13:00

General Comments:

Activity 1 - Task planning and system design changes – see **TASK_PLAN_76543_QB00700_BRANCH_Q.pdf .**
Planned work v achieved work during session #1 was a complete success.

Activity 3 - System Design

(1) List of possible input & output hardware options and microcontrollers.

The microcontroller options, including development boards I have considered are the Atmel ATMega328P on the Arduino UNO board and the Microchip PIC16F1937 on the Matrix HP4988 board. I am familiar with both boards and their software development environments.

The Matrix HP4988 development board comes with a range of keys and an LCD display as standard.
The Arduino UNO has no integrated keys or LCD, however the D1 Robot shield provides access to keys and an LCD display.

Arduino UNO + D1 Robot shield:

Arduino UNO with the
D1 Robot shield fitted.

Matrix HP4988:

Matrix HP4988 has keys and LCD available as standard.

Additional hardware:

Both the Arduino UNO and Matrix HP4988 need extra hardware in order to implement a bright LED and piezo sounder. In both cases this hardware can be built on breadboards and attached to the microcontroller development boards. In fact the same hardware is suitable for both systems.

L-813SRC-B - LED, Red, Through Hole, T-3 (10mm), 20 mA, 1.85 V, 660 nm

Kingbright

Manufacturer:	KINGBRIGHT
Manufacturer Part No:	L-813SRC-B
Order Code:	1142466
Technical Datasheet:	(EN)

See all Technical Docs

The L-813SRC-B is a 10mm super bright red Solid State LED Lamp with water clear lens. The super bright red source colour device is made with gallium aluminium arsenide red light emitting diode.

PKM13EPYH4000-A0 - Transducer, Sounder, Sounder, 25 V, 70 dB

muRata
INNOVATOR IN ELECTRONICS

Manufacturer:	MURATA
Manufacturer Part No:	PKM13EPYH4000-A0
Order Code:	1192513
Technical Datasheet:	(EN)

Suggested replacement for 4136275

Comparison of Microcontrollers and development boards

Parameter	Miscocontroller		Comments
Micro-controller	ATMega328P Arduino UNO	PIC16F1937 Matrix HP4988	Both are 8 bit CPUs with Harvard type architectures.
Flash ROM (Bytes)	32K	16K (8K Words)	The Atmel chips use byte wide ROM for program storage whereas the PIC chips use a 14 bit wide ROM for program storage to match their 14 bit wide instructions.
RAM (Bytes)	2K	512 bytes	I expect to need no more than 100 bytes of RAM for my solution, so both microcontrollers are suitable in this respect.
EEPROM	1K	256 bytes	I do not intend to use EEPROM in my solution.
Max Clock Rate	20MHz	32MHz	The Atmel chips mostly run one instruction per clock cycle whereas the PIC chips need 4 clock cycles to execute one instruction. However, the PIC chips execute single 14 bit wide instructions, whereas the Atmel chips execute multiple byte wide instructions. On balance, the microcontrollers have similar performance when it comes to speed.
I/O Pins	23	36	I expect to use approximately 20 I/O pins for my solution, so either microcontroller is suitable when it comes to I/O capability.
Max source / sink current through a port pin	Source 20mA	Source 25mA	Although the PIC16F1937 can source and sink more current on a port pin, I don't think that the difference is significant.
	Sink 20mA	Sink 25mA	
ADC	8 channels x 10 bit ADCs	14 channels x 10 bit ADCs	I do not intend to read analogue voltages for my solution, so the ADC channels will not be required.
8 bit Timers	2	4	I expect to use one or two timers for my solution, so either microcontroller is suitable.

16 bit Timers	1	1	See comment for 8 bit timers
UART	YES	YES	I do not intend to use serial communications in my solution, so the UART will not be required.
I^2C	YES	YES	I do not intend to use I^2C peripherals in my solution, so the I^2C facility of the microcontrollers will not be required.
SPI	YES	YES	I do not intend to use SPI peripherals in my solution, so the SPI facility of the microcontrollers will not be required.
D10 Robot shield switches & LCD	Yes	N/A (Arduino specific)	Advantages: Only uses one I/O Pin on microcontroller (ADC input) for all 6 key inputs. LCD has I/O pin controlled backlight.
			Disadvantages: Only 6 keys available
eBlocks EB083 Matrix board (and EB006 multiprogrammer)	N/A (Matrix HP4988 specific)	Yes	Advantages: 16 switches, LCD, Seven segment LED display, LDR etc on one interface board.
			Disadvantages: Some combinations of I/Os are mutually exclusive – i.e. port B, bits 0 to 5 not available for switch inputs when the LCD is in use.
Development Environment – Arduino IDE	Yes	N/A (Arduino specific)	Advantages: Freeware and lots of libraries and example code for all sorts of hardware.
			Disadvantages: No debug or simulation facility (can be done using AVR Studio 7 and AVR JTAG ICE, adding extra complexity to development). No integrated documentation generation – needs to be done outside of development environment.

Development Environment - Flowcode 7	N/A Although there is a Matrix eBlocks board based on the Arduino UNO	Yes	Advantages: Automatically generates suitable design documentation (flowcharts) as part of the development. Good debug and simulation facilities with single stepping, breakpoints and variable watches.
			Disadvantages: Code generators like Flowcode do not always produce the most efficient code, so the solution will use more ROM than a hand coded program. The Flowcode program may also run slower for the same reason. My solution is not code or time critical, so this is not an issue.

Microcontroller and development board selection:

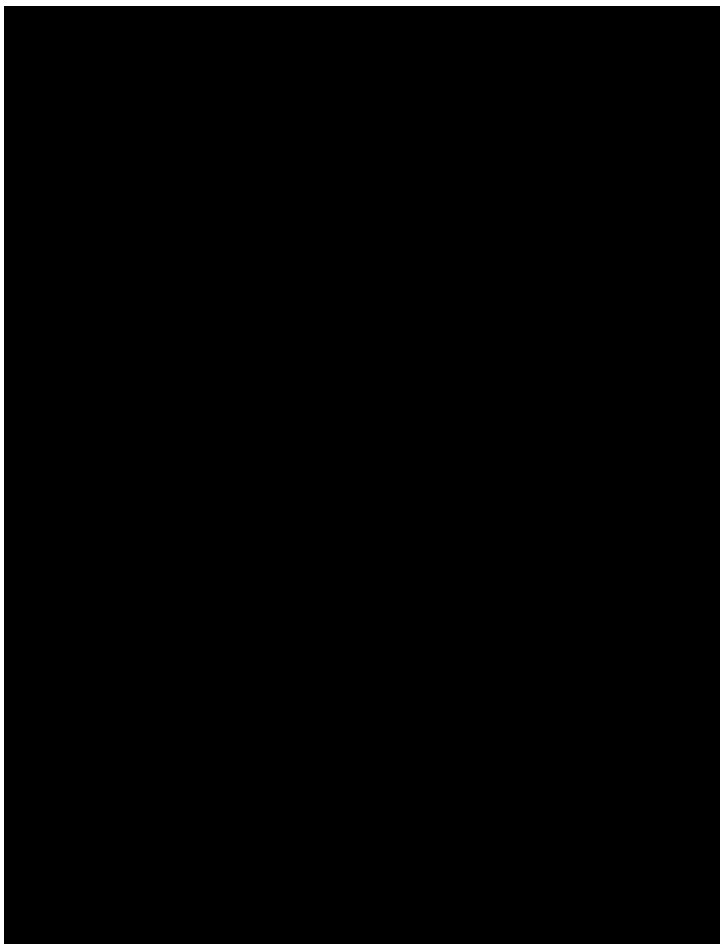
I have decided to use the Matrix HP4988 because as it has a better development environment with debugging facilities including stepping through code, breakpoints and watches on variables. The Flowcode 7 software environment also provides good built in documentation facilities.

The Matrix HP4988 also comes with more keys than the Arduino UNO option, allowing a clearer user interface for my device.

Outline System Plan:

Action	**Hardware required**	**Software implementation**	**Risks**
Read keys	Matrix HP4988	Flowcode “Input”	Keys may need debouncing
Display time remaining, welcome message etc	Matrix HP4988	Flowcode component macros for LCD	None
Measure time	Matrix HP4988	Flowcode timer event	Selection of correct prescaler and timer reload values
Flash bright LED	LED and resistor connected to Matrix HP4988	Flowcode “Output”	Selection of correct current limiting resistor value and power rating
Sounding Piezo transducer	Piezo and resistor connected to Matrix HP4988	Flowcode “Output” and possibly a timer event	Selection of correct current limiting resistor value and power rating

Hardware design for the additional device hardware:

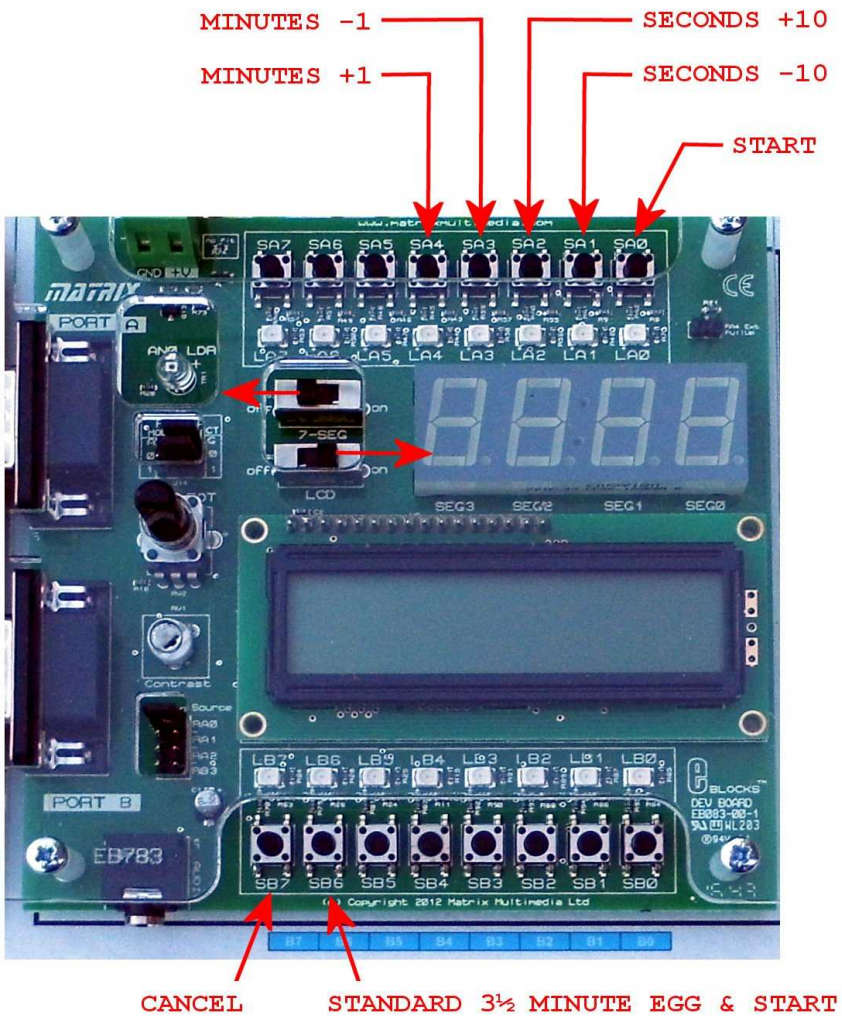


From the Piezo data sheet, it has a capacitance of 5.5nF. As the drive pins (Port pin 2 and pin 3) change polarity across the piezo, this capacitance will cause a current to flow through R_PIEZO for a short time until the voltage across the capacitor has reversed polarity. I need to limit the current flowing through the microcontroller pins to 20mA. R_PIEZO = ($5V_{(pin2\ to\ pin3)}$ + $5V_{(piezo\ charge\ voltage)}$) / 20mA = 500Ω and the nearest higher preferred value is 680Ω with a power rating of 680Ω x $(20mA)^2$ = 272mW. However, this pulse of current will only flow for a short amount of time so I will use a 0.25W resistor.

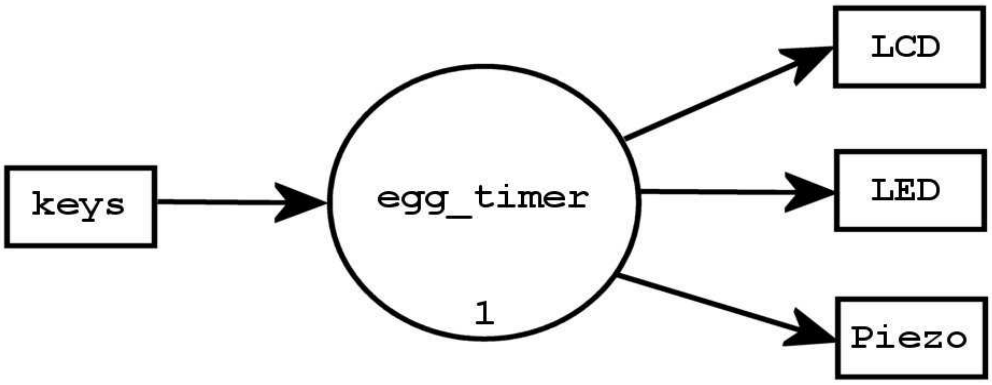
From the LED data sheet, the minimum voltage drop across the LED is 1.85V @ 20mA, leaving 3.15V across R_LED. If I use a 180Ω resistor for R_LED, the LED current will be 3.15V / 180Ω = 17.5mA which is acceptable. The power rating of this resistor must be at least $(3.15V)^2$ / 180Ω = 55mW, so I will use a 0.25W resistor.

Additional hardware connected to Matrix HP4988 board

Matrix HP4988 board key functions:



Top level software design (context diagram):



Issues encountered and solutions with justification:

(1) There is no bright LED or piezo sounder on the Matrix HP4988 board as standard, so I designed the extra hardware needed. I also researched the possible connection options to integrate my hardware to the HP4988 board.

Action list for the next session:

(1) Continue with software design.
(2) Modify hardware design if necessary.
(3) Assemble the hardware.
(4) Test the hardware with some simple test software.