Project 3:
Learning the Development Environment – The Next Step

EE/CSE 474 EMBEDDED MICROCOMPUTER SYSTEM

Contributor: Gaohong Liu, Shawn Xu, Yongqin Wang
Instructor: James Peckol, Daniel Predmore, Christy Truong, Jonathan Hamberg

University of Washington
Department of Electrical Engineering

Abstract

In Lab3, we implemented a system that the kernel in System Control Panel can dynamic control and schedule task and assign resources based on different phase of process while under the control of System Control, Peripheral System can measure data from off-system devices and send to back to system and waiting for System Control to assign task to compute and to output. Meanwhile, by using Display Panel, users can directly control the whole system to measure the items they need to display and stay in top menu if they would like to keep system sleep.

Table of Contents

1. Introduction

This project focused on the design of basic system architecture, data model, control flow and data exchange and operation control for a low cost, portable, medical monitoring system. The system is designed to dynamic control whole process, and peripheral could measure data and system control could schedule task to simulate four kind of information (blood pressure, body temperature, pulse rate and battery status) and the display of six kinds of information (blood pressure, body temperature, pulse rate, battery status and warning/alarms) and store. We used UML - Use Cases, Functional Decomposition, Sequence Diagrams, State Charts, and Data and Control Flow Diagrams to design the architecture of the system, C language to implement functionality, Codeblock IDE to debug and finally loaded the final product to the Arduino ATMega2560 and Uno.

2. Discussion of the Lab

Design Specification

Overall Summary

The product developed is the medical monitor device that can support many input and output peripherals. After first stage in Lab2, current stage of such product is to have the ability in three parts. First is that Peripheral system could measure on human body temperature, pulse rate, blood pressure and device status from outside world by connecting with other devices like pulse generator. Second part is that we have to implement communication protocol and channel to send data from Peripheral measurement system to Control system to process, display and output(display). Due to system limits, we have to use Peripheral system to measure and can not use Control system to implement measurement. Last part is that after receiving measured data, it could dynamic schedule task including measurement, calculate, warning alarm, display and status which can produce corrected result of human body temperature, pulse rate, blood pressure and device status, monitor on human body temperature, pulse rate, blood pressure and device status, produce warning/alarm based on some critical values of corresponding information, display body temperature, pulse rate, blood pressure, device status and warning and alarm information on TFT display.

To achieve the desired functionality of such system, the system is divided into two subsystems: Uno served as Peripheral and ATMega2560 served as System Control. In addition TFT display panel served as human interface where users should use TFT panel to operate. It is responsible for properly displaying three kind of information on the TFT display: annunciation, status and warning / alarm. Uno and ATMega are responsible for receiving all kinds of human health data, process them, store them and use TFT display panel to output.

Use Cases

This is the topology of our fully implemented application. Two parts consist of our system. ATMega2560 and Uno. Served as Control or Kernel, ATMega2560 has a dynamic control kernel which can control the status of whole system. Connecting with pulse generator, Uno works as a measurement tool and it receives data from outside world and send them to ATMega to

processes and displays.

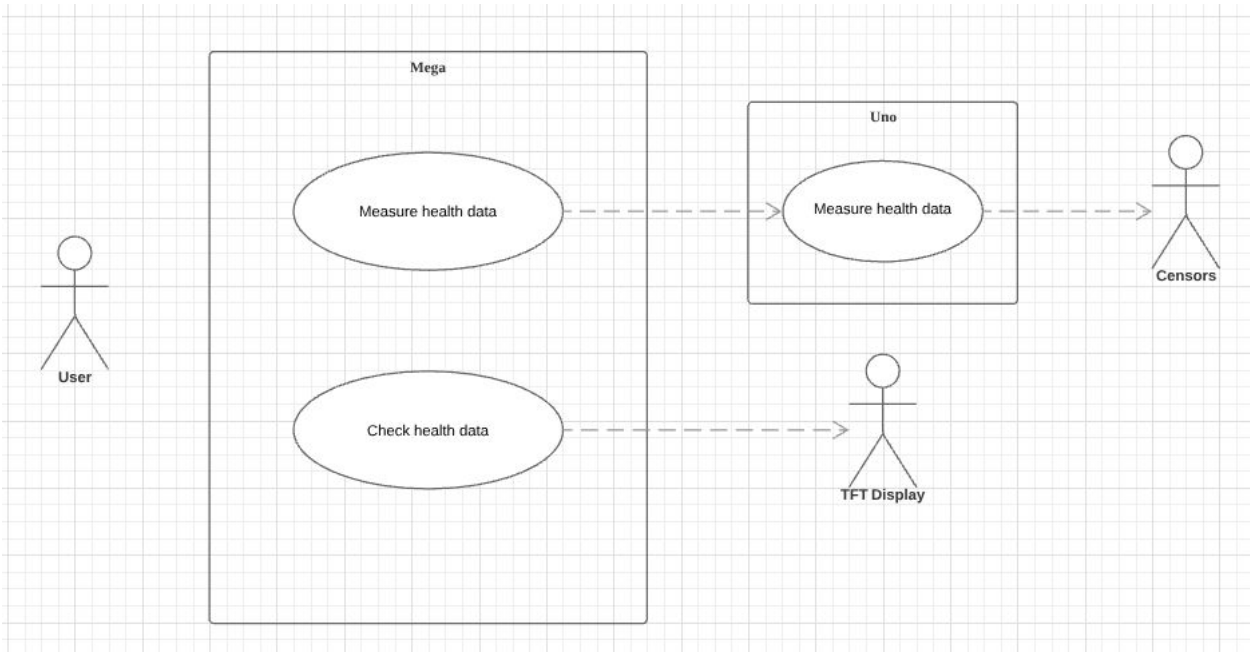

figure 1 Use Case Diagram

Interface

User to Computer Interface

Input:

Annunciation and Measurement Menu

Output:

Measurement Item

Body Temperature, Blood Pressure, Pulse Rate

Annunciation

Blood Pressure, Body temperature, Pulse rate, Battery Status

System Process Interface

Input

Measurement:

Blood Pressure, Body temperature, Pulse rate

Output

Display:

Blood Pressure, Body temperature, Pulse rate,

Status:

Battery state

Alarms:

Temperature, Blood pressure, Pulse rate too high

Warning:

Temperature out of range, blood pressure out of range, pulse rate out of

range

Side Effect

Before Annunciation, it will display two Green Box. One is menu and the other one is Annunciation.

When choose Annunciation, display background is Black. In our display panel, all the data are in a Green ASCII Drawing Box. The first line is our product's name and the last line is our greeting.

When user's data are in normal range, data will display in Green, otherwise data that are out of range will display in Red. If choose acknowledgement, it will display Yellow and return back to Red in 5 seconds.

Pseudo English description of algorithms, functions, or procedures

Main function is the interface where users could choose the items they would like to measure and make system awake to work or sleep

```
main():
      If pressure on screen is in valid range:
            If user is in top menu:
                  Find the range users touch
            If user is in measurement menu:
                  If user choose return:
                        Return to top menu:
                  If choose to measure temperature:
                        Measure temperature
                  If choose to measure pulse:
                        Measure pulse
                  If choose to measure pressure:
                        Measure pressure
            Else if:
                  If user is in annunciation menu:
                        Call dynamic scheduler
```

intToChar function is used to change raw data to ASCII Code. Because in reality, human body temperature will not excess 100 celsius degree, and human blood pressure and battery status will not excess 3 digits, we assume all of our data at most have 3 digits.

```
intToChar(unsigned char* result, int RawData):
      first char = Rawdata / 100 + 48
      second char = RawData % 100 / 10 + 48
      last char = RawData % % 10 + 48
```

Measure function is used to "measure" our input.

```
measure (void* data) :
      // Temperature Measurement
      if temperature > 50:
            !tempUp
      else if temperature < 15:
            tempUp
```

```
if tempUp:
      if times is even:
            temperauteRaw+=2
      else:
            temperatureRaw-=1
else:
      if times is even:
            temperauteRaw-=2
      else:
            temperatureRaw+=1

// Pulse Measurement
if pulse > 40:
      !pulseUp
else if pulse < 15:
      pulseUp
if pulseUp:
      if times is odd:
            pulseRaw+=3
      else:
            pulseRaw-=1
else:
      if times is odd:
            pulseRaw-=3
      else:
            pulseRaw+=1
// Systolic Pressure Measurement
if systolic measurement is Not Done:
      If time is even:
            systolicPressRaw+=3
      Else:
            systolicPressRaw-=1
      If systolicPressRaw > 100
            systolicMeasurement is Done
            systolicPressRaw = 80 // Reset
// Diastolic Pressure Measurement
else:
      if time is even:
            diastolicPressRaw-=2
      else:
            diastolicPressRaw+=1
      if diastolicPressRaw < 40
            Systolic pressure measurement is not Done
            diastolicPressRaw = 80 // Reset
if time is even:
      time is odd
else:
      time is even
```

compute function is used to translate raw data to human readable data

```
Compute (void* data) :
      Temp = 5 + 0.75 * temperatureRaw
      Systo = 9 + 2 * systolicPressRaw
      Diasto = 6 + 1.5 * diastolicPressRaw
      Pulse = 8 + 3 * pulseRateRaw
      intToChar(tempCorrected, temp)
      intToChar(sysPressCorrected, systo)
      intToChar(diastolicPressCorrected, diasto)
      intToChar(pulseRateCorrected, pulse)
```

displayF function is used to configure human readable data on display panel

```
displayF(void* data):
      print(temperature)
      print(distolic pressure)
      print(systolic pressure)
      print(pulse rate)
      print(battery status)
```

statusF function is used to calculate the remaining battery energy

```
statusF(void* data):
      if battery state > 0:
            battery state --
```

Warning alarm function is used to set flags of all the data which can indicate whether data are out of range

```
warningAlarm(void* data):
      if temperature < 36.1 or temperature > 37.8 :
            tempOutOfRange
            if temperature > 37.8:
                  tempHigh
      else:
            !tempOutOfRange
            !tempHigh
      if systolic Pressure > 120 or diastolic Pressure < 80:
            Blood Pressure Out Of Range
            if Diastolic Pressure < 80:
                  Diastolic Pressure Out Of Range
            else:
                  !Diastolic Pressure Out Of Range
            if Systolic Pressure > 120:
                  Blood Pressure is High
            else:
                  !Blood Pressure is High
      else:
            !Blood Pressure Out Of Range
            !Diastolic Pressure Out Of Range
```

```
        !blood Pressure is High

    if Pulse Rate < 60 or Pulse Rate > 100:
        Pulse Rate Out Of Range
        If (Pulse Rate < 60):
            Pulse Rate is low
    else:
        !Pulse rate Out Of Range
        !Pulse rate is low

    If battery state < 20:
        Battery state is low
```

Scheduler function is coupled with issue function where scheduler schedules task and send their information to issue. Then issue function issueses them.

```
    Scheduler():
        Head = NULL
        Tail = NULL
        issue_count = 0
        issue(measure count, measure period, 0, blocks)
        issue(compute count, compute period, 1, blocks)
        issue(warning count, warning period, 2, blocks)
        issue(display count, display period, 3, blocks)
        issue(status cpimt, status period, 4, blocks)
    issue(count, period, which, blocks):
        if count equal to 0:
            *blocks[which].myTask)(blocks[which].taskDataPr
        if count equal to period:
            Count = 0
        else:
             count ++
```

Communication is shared by both ATMega2560 and Uno where they have to have same amount of parts to receive and transmit data back and forth.

```
    Uno:
        Receive start bit
        Receive measure function number
        Receive end bit
        Send start bit
        Send measured data
        Send end bit
    ATMega2560:
        Send start bit
        Send measure function number
        Send end bit
        Receive start bit
        Receive measured data
```

```
            Receive end bit
```

DoublyLinkedList is our data structure to implement dynamic scheduler and RingBuffer is data structure to store Warning Data

```
      Class DoublyLinkedList:
            insert(TCB* node):
                  If NULL == head:
                        Head = node
                        Tail = node
                  Else :
                        Tail.next = node
                        node.prev = Tail
                        Tail = node
                  Return
`           deleteNode(TCB* node):
                  If (NULL = head):
                        Return
                  Else if (head == tail):
                        Head = NULL
                        Tail = NULL
                  Else if (tail == node):
                        Head = head.next
                        Node.next.next = NULL
                        Node.next = NULL
                  Else
                        Node.prev.prev = node.next
                        Node.next.prev = node.prev
                        Node.prev = NULL
                        Node.next = NULL
                  Return
      Call RingBuffer:
                  Put (char val, int size):
                        inpulseBuffer[bufHead] = val
                        If (lead == 1):
                              If bufHead + 1 >= size:
                                    Lead = 0
                                    bufHead = 0
                              If bufTail = 0:
                                    bufTail = 1
                                    Lead = 0
                              Else:
                                    bufHead++
                        Else:
                              If bufHead + 1 == bufTail:
                                    If bufTail + 1 == size:
                                          bufTail = 0
                                    Else:
```

```
                    bufTail++
            Else:
                bufHead++
```

Execution Time

For the first run, due to the cold start:

- Measure: 16μs
- Compute: 236μs
- WarningAlarm: 88μs
- Display: 718676μs
- Status: 12μs

For the rest run (almost all of rest run):

- Measure: 8μs
- Compute: 8μs
- WarningAlarm: 80μs
- Display: 8μs
- Status: 8μs

Annunciation will take 2 seconds to process.

However, due to the hierarchy of built-in cache, conflict miss or capacity miss, the time is varied. The probability to have such situation is pretty small.

Timing constraints

Due to the refresh rate of the screen and computation rate, the delay between each display should be larger than 10ms

Error handling

We try to give all data a flag to indicate their behavior. Out of Range or not, High, Low. The reason we have so many flags is we can simplify our logic to avoid losing significant logic in controlling Warning Alarm and Measure function.

In addition, our user interface has forced users to operate under our process procedure where they have to choose things to measure, and return to top menu and press annunciation to start, otherwise it the whole system will not awake to start working.

Software Implementation

Functional Decomposition Diagram for the System Control

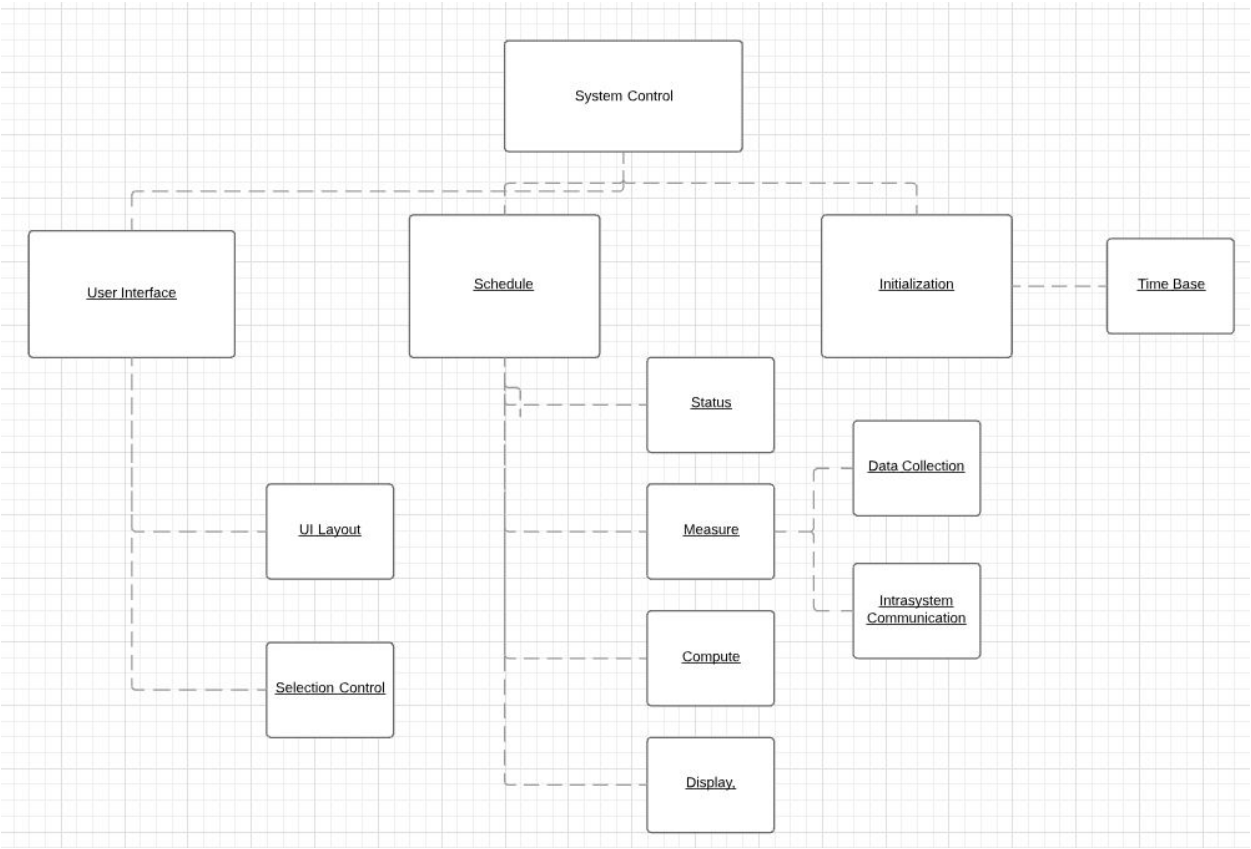

Figure 2 System Control Functional Decomposition Diagram

System Control Task/Class Diagrams

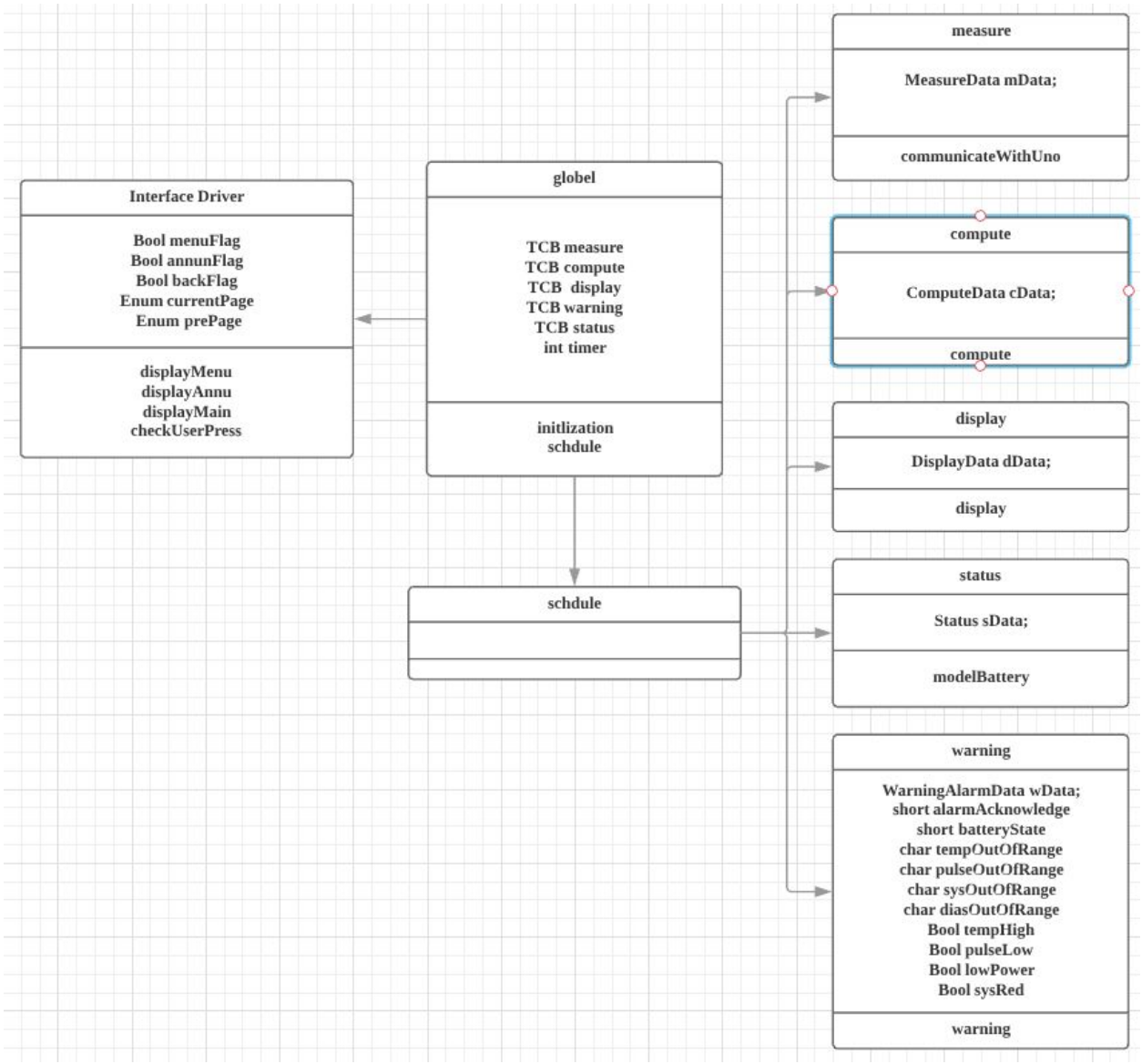

Figure 3 System Control task/class diagrams

System Control Activity Diagrams

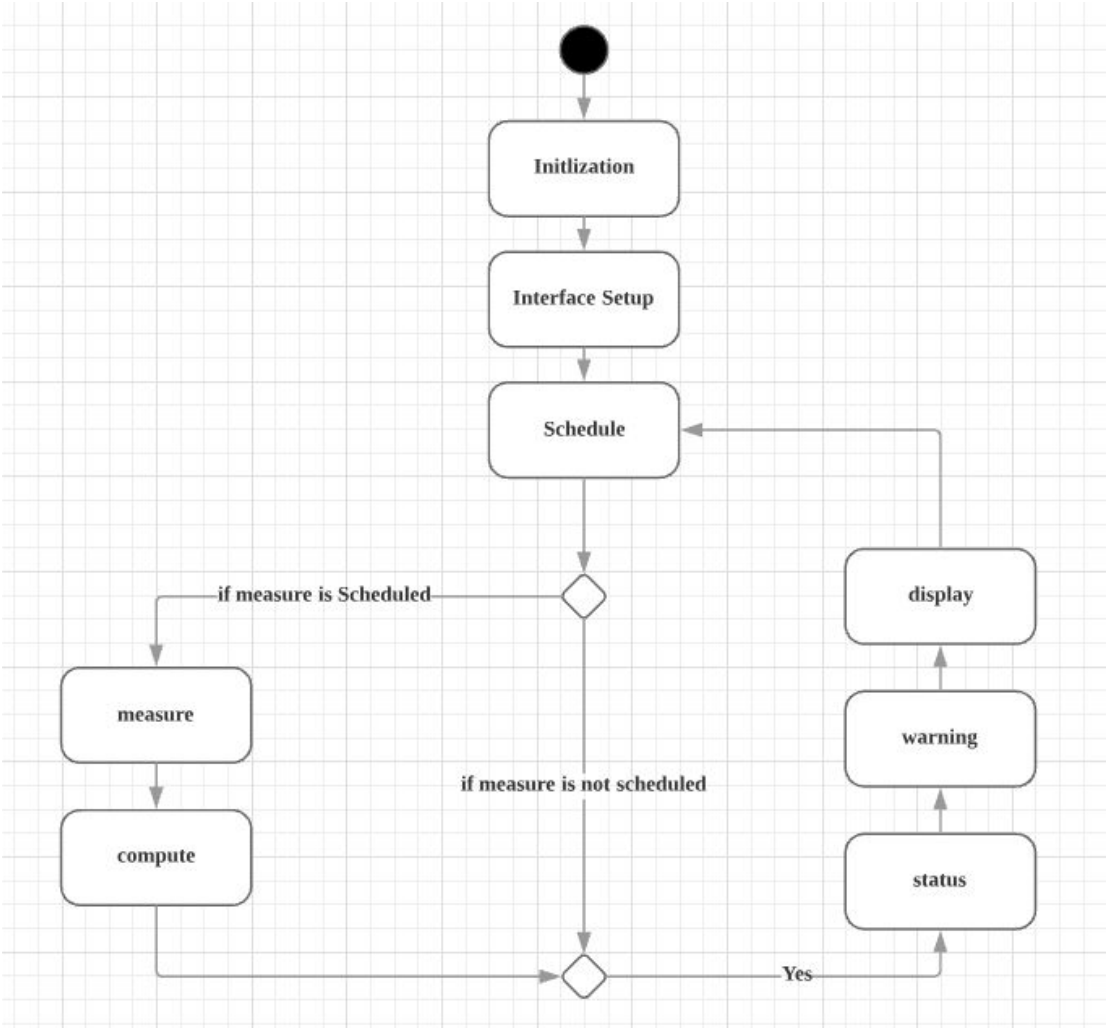

Figure 4 System Control Communication Task Diagrams

Functional Decomposition Diagram for the Peripheral Subsystem

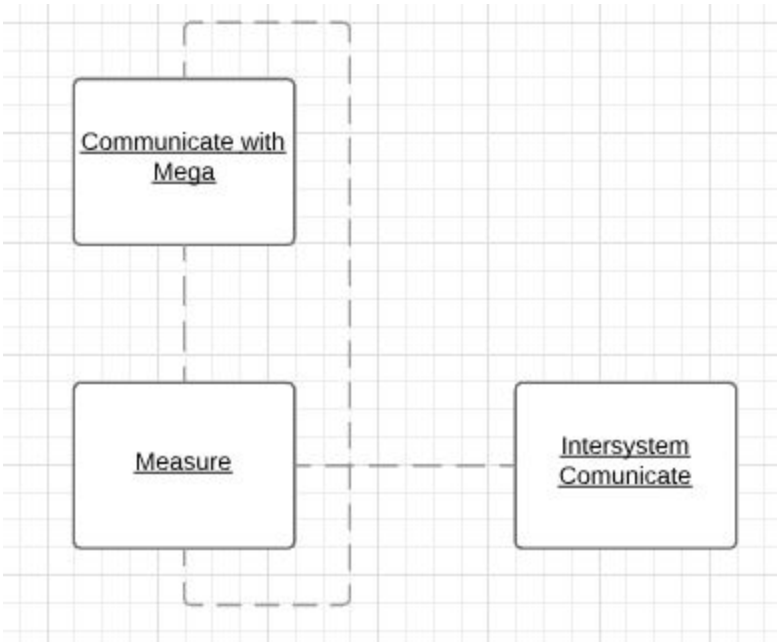

Figure 5 Peripheral Subsystem Functional Decomposition Diagram

Peripheral Subsystem Task/Class Diagrams

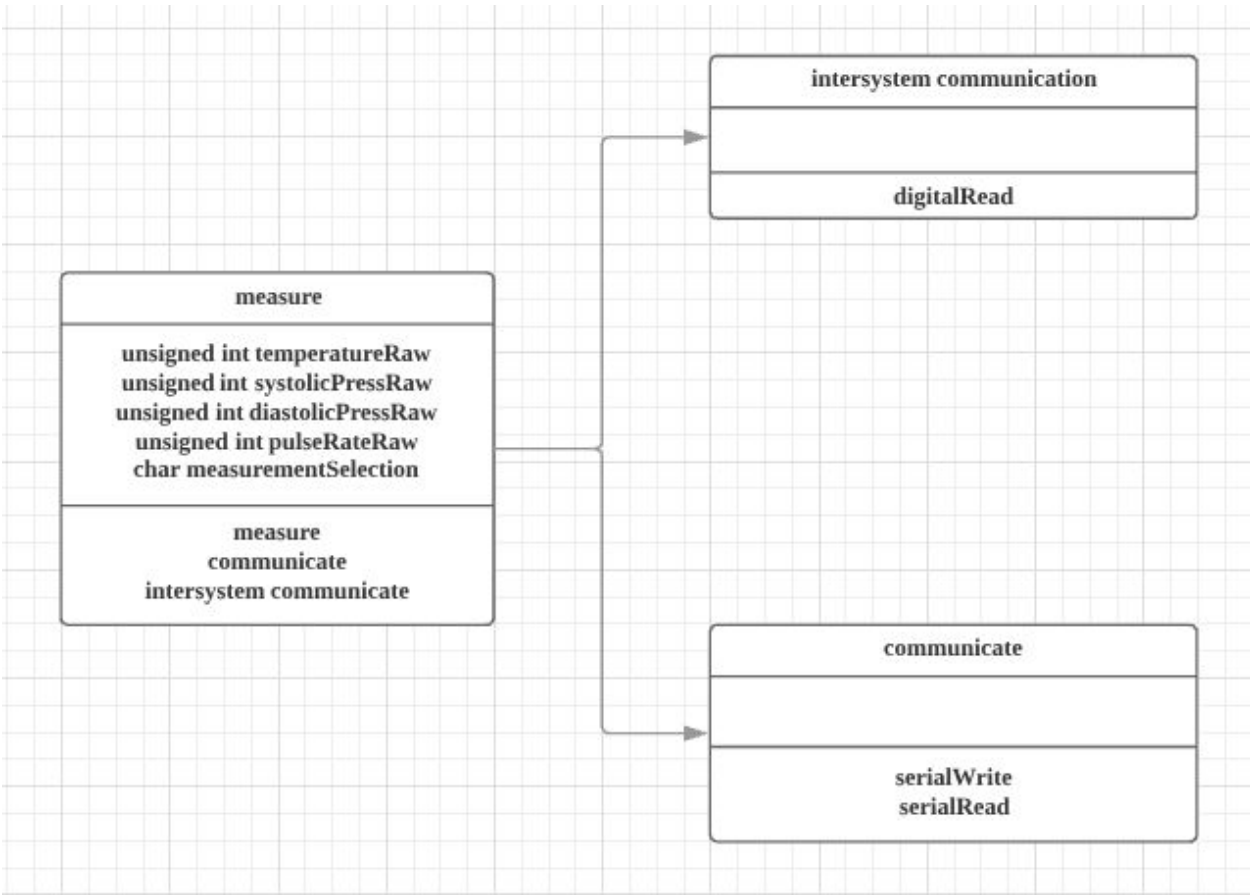

Figure 6 Peripheral Subsystem task/class Diagrams

Peripheral Subsystem Activity Diagrams

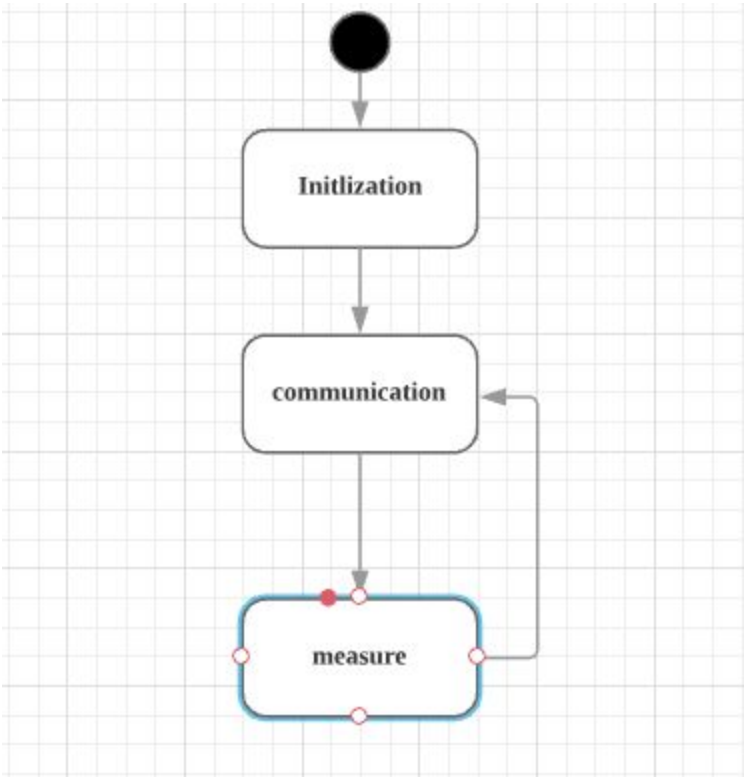

Figure 7 Peripheral Subsystem Activity Diagrams

3. Presentation, discussion, and analysis of the results

Overview:

The final product has four critical checkpoints which need to be confirmed to ensure the correctness of the implementation. First, user interface must be usable. Second, user interface must be able to record the tasks selected and the system must be able to ensure the corresponding tasks are set to be executed. Third, pulse rate reading is now modeled by analog input from signal generator instead of modeling internally in the previous lab. Thus, the pulse rate reading must be correspond to signal generator input and varies accordingly when the frequency of input signal varies. Finally, the warning/alarm system are updated with new color codes and new rules of display. The rest of rules from previous lab remains in effect. Since there is no alternation made on those parts of implementation, we ensure the correctness by inspection and the result will not be included in this lab report. Please refer “EE474 Lab2 report” for more information.

User Interface:

The display function improved its functionality by adding user interface that grand user more control power on the system. Because of the rapid prototype strategy we employed in the development process, each tab will be position-holding colored blocks with correct function implemented and obviously lacking intuition and aesthetics. More informative tabs will be added

later in this project.

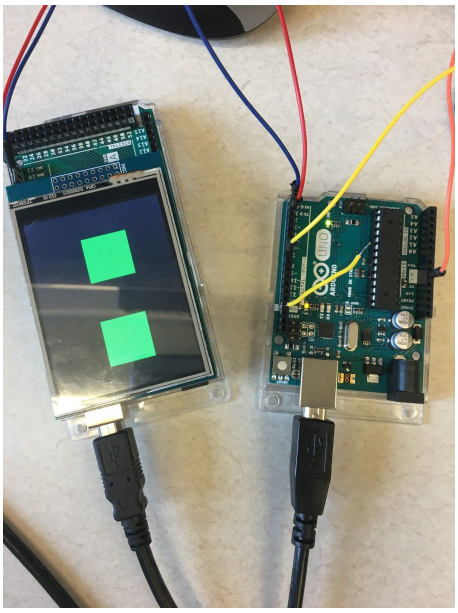

Figure 8 Main Page

The display task supports two modes: Menu and Annunciation. As the figure 8 demonstrates, two green blocks represent menu and annunciation. The top block represent menu and the bottom block represent annunciation. Press on those two blocks will take user to its corresponding section.

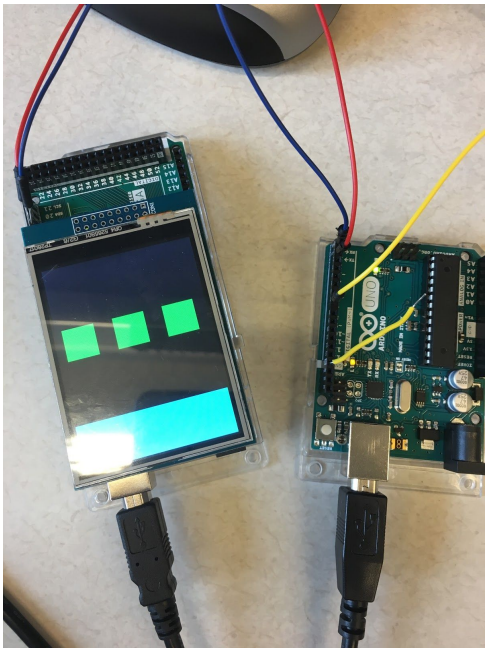

Figure 9 Menu Page

Click on the top block (menu), user will see three square boxes and a blue rectangle box demonstrated by the figure 9 above. The leftmost green box is the switch for body temperature measurement. The middle box is for blood pressure measurement. The right box is for pulse rate

measurement. When measurement is selected, the box will be green. If it was tabbed to unselected, the box will be red. As figure above showed here, the default selection is all the measurement. The blue box is the “back” button which let user return to main page.

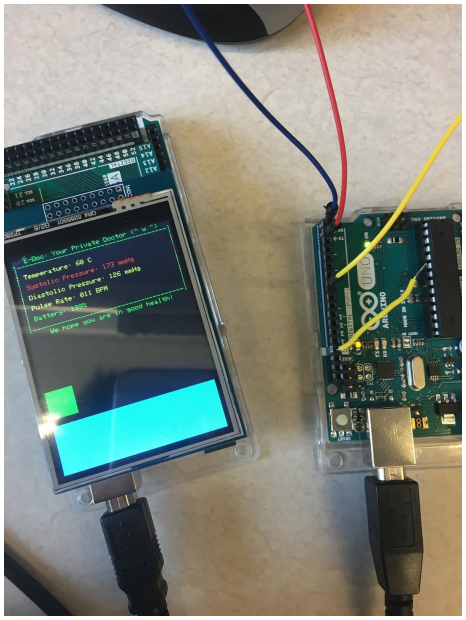

Figure 10 Annunciation Page

Click on the bottom block (annunciation) on the interface showed by figure 8, user will see the annunciation page similar to previous lab. In this page (figure 10), the user interface will only display data of chosen measurements. The green box in the left corner is the acknowledge button which let user to acknowledge alarm and the blue box again is the back button for return to the main page.

<u>Tasks Selection:</u>

The system must allow user to select the measurement to execute and execute selected measurements. In this implementation, the task selection button demonstrated by Figure B enable user to select between measurements. When measurement is selected, the box will be green. If it was tabbed to unselected, the box will be red.

- When all three measurement are selected (figure 9), the annunciation will display the data for all three measurement(figure 10).

- When body temperature measurement is unselected(figure 11), the annunciation will not display the data for body temperature measurement(figure 12).

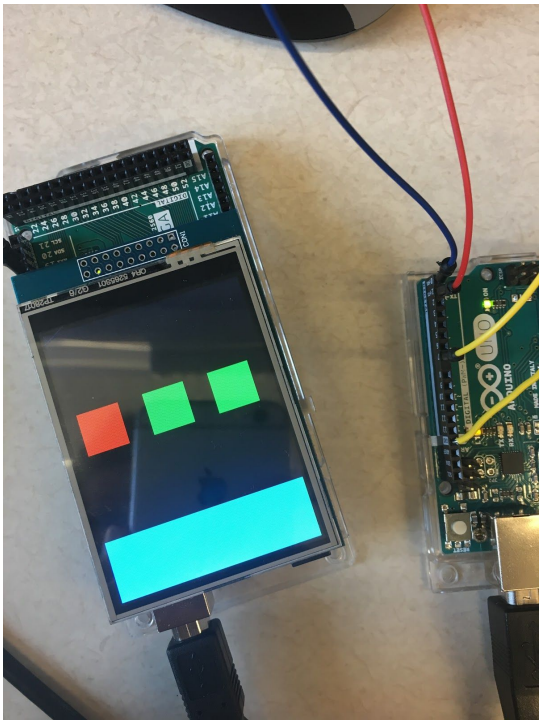

Figure 11 Unselect Temperature Measurement

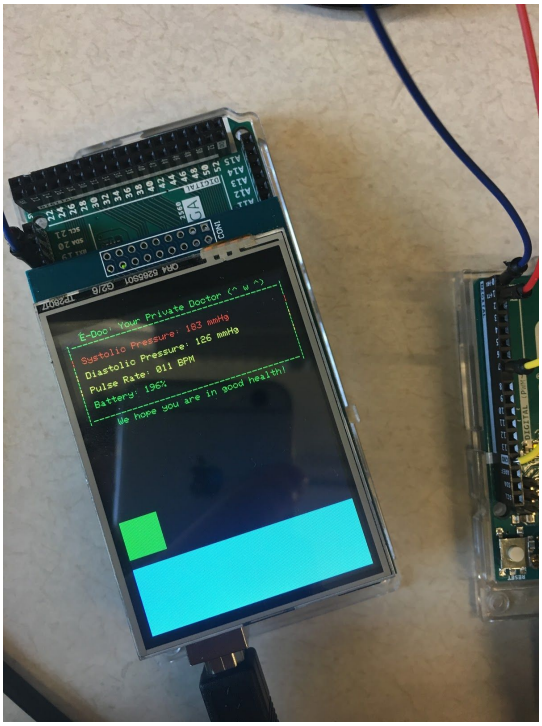

Figure 12 Annunciation Page with Temperature Unselected

- When body temperature measurement and blood pressure are unselected(figure 13), the annunciation will not display the data for body temperature measurement and blood pressure (figure 14).

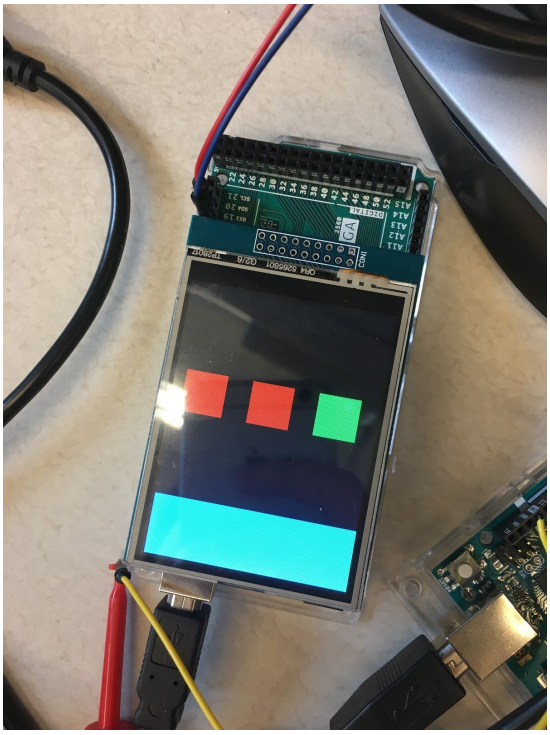

Figure 13 Temperature and Blood Pressure Unselected

Figure 14 Annunciation Page with Temperature and Blood Pressure Unselected

- When all three measurements are unselected, the annunciation will only display battery

status(figure 15).

Figure 15 Annunciation Page with All Measurements Unselected

Compare the expected outputs with the actual outputs, we confirm the task displayed is indeed corresponding with task selected.

Pulse Rate Reading:

In this lab project, pulse rate data is modeled using the input from signal generator. We must make sure the pulse rate reading is corresponding to the signal generator input frequency and the reading change with input change correspondingly.

Figure 16 Signal Generator at 1kHz

Figure 17 Annunciation Page with Corresponding Pulse Rate

Figure 18 Signal Generator at 100kHz

Figure 19 Annunciation Page with Corresponding Pulse Rate

Compare the two sets of figures (figure 16,17 / figure 18, 19) above, we confirm the pulse rate measurement indeed corresponding to input frequency with lower frequency correspond to lower pulse rate reading and higher frequency correspond to higher pulse rate reading. We can also confirm the pulse rate reading change with input frequency accordingly.

<u>Warning/Alarm:</u>

Serval modifications are made on warning/alarm system. We need to ensure four points: 1) data is displayed in orange if it is out of range. 2) systolic blood pressure is displayed in red if it is over 20% out of range 3) when systolic blood pressure is displayed in red, it will changed to orange if acknowledge button is pressed. 4) If the systolic blood pressure remains out of range for more than five measurements, the systolic blood pressure data will be displayed in red again.

- Data displayed color:

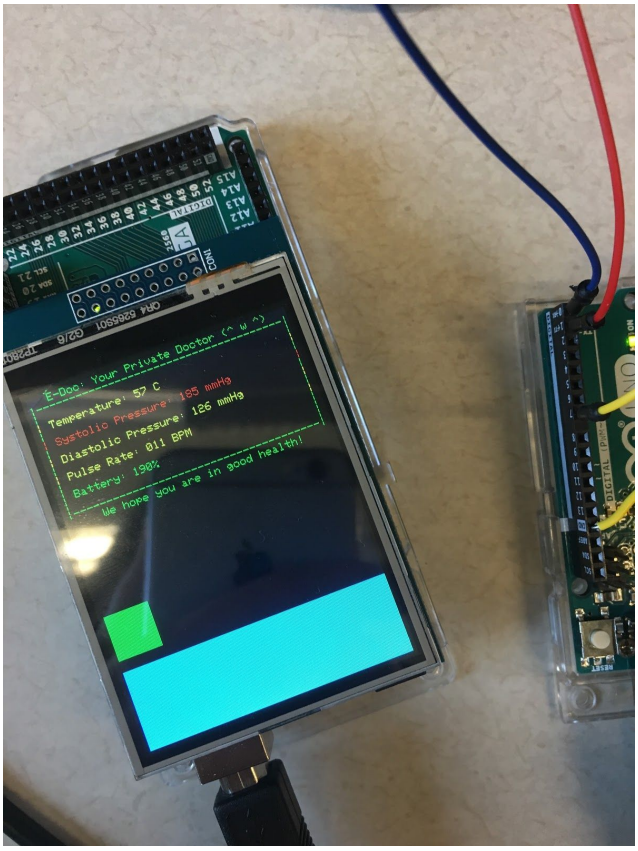

Figure 20 Annunciation Page with Correct Warning Color

Demonstrated by figure 20, temperature (above 36.1), diastolic pressure (above 80) and pulse rate data (below 60) are all in orange because they are out of range. The systolic pressure are in red because it is out of range by 20% (42% in this case).

- Acknowledge Pressed:

Figure 21 Annunciation Page with Acknowledge Button Pressed

Compare with figure 20, figure 21 showed that after acknowledge button (green box) is pressed, the systolic pressure display changed from red to orange despite the value is still out of range more than 20%.

- Five measurement after acknowledge pressed:

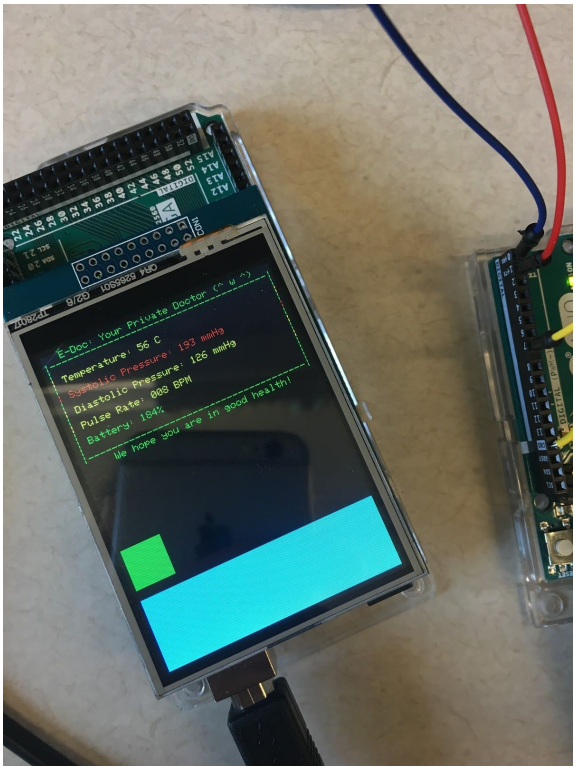

Figure 22 Annunciation Page Five Circle after Acknowledge Button Pressed

Compare with figure 21, figure 22 showed that after acknowledge button (green box) is pressed for five second, the systolic pressure display changed from orange to red since the value is still out of range more than 20%.

Compare the expected outputs with the actual outputs, we confirm the warning/alarm system modification worded as expected.

4. Analysis of errors needed to be resolved

- Measure Flag: We use UI mode and blocking timer to decide whether to put a task into task queue instead of setting up a global flag. Although they all achieve the same effect and final presentation are correct, we will change our implementation in next lab.

- Task Timer: We setup a timer for each timer to achieve time blocking for each tasks, but we should only have one global timer. We will change our implementation so that blocking timers will rely on global timer.

- User Interface: During demo, we did not put text on screen to indicate function for each tab, we will add text indication.

- Text Color: We use yellow to display acknowledge warnings, which should be orange. We will change yellow to orange in next implementation.

5. Analysis of errors encountered

1. Communication inconsistance.
   During delevelopment, we found out sometimes received data in ATMega are not the same as we sent in UNO when we are printing data on UNO's console.
   Cause: UNO only have one serial port, printing on console will disturb serial communication.
   Fixation: We delete will print statement on UNO's side when we are doing serial communication.
2. # of TCB is always 0
   Causes: we assign execution pointer prior to we making task queue. Thus, execution pointer is always pointing to pointing to null.
   Fixation: Assign execution pointer after task queue is made.

6. Test

Test Plan

* Tasks the same as those in lab2 will be tested using the same plan and cases as we did for lab 2. Please refer "EE474 Lab2 report" for more information.

Data structures

LinkedList

We need to verify that linked list we created will can handles inserts and deletes appropriately and it could also handle edge cases correctly.

UNO side

In order for our system to work properly, we need to at first verify UNO side works correctly.

1. We need to verify that the measure function still produces the same result as we saw in project
2. We need to verify that data UNO measured for pulse rate would change to appropriate value based on frequency of incoming square wave.
3. UNO are able to transmit data to ATMega through TX and RX port.
4. Received data on ATMega side are identical with those on UNO side.

ATMega side

Secondly, we need to verify that new features added like interchangeable UI works appropriately

1. We need to verify that UI interface could change appropriately based touches.
2. We need to verify that global flags indicating which measurement to make will are synchronous with what user selected on screen.
3. We need to verify that acknowledge flag will response to a touch to specific region in annunciation mode and it will be clear after a period of time or warning has been cleared.

4. We finally should verify that the number of TCB in the tasks in queue should be the same as what we desired.

If those tests are passed, we can safely admit our code behave as we desired.

Test Specification

Data structure
We test this data structure using code block and we use printf to print out what is inside of linked list every time we do a modification of the list. We verify printed log against what we expected.

DoubledLinkedList
This is a data structure that containing a numbers of TCB blocks which contains their own data and prev/next pointers which point to its preceding/following TCB.
Head & Tail
Global variable head and tail should always point to the start/end of the linked list.
If there is no element in a DoubleLinkedList, those two should point to NULL.
If there is only one element in the doubledLinkedList, head and tail should point to the same block.

Insert function
This function should take in a pointer to a tcb block and append to the end of the doubledLinkedList.

Delete function
This function takes a pointer to a tcb which should be in doubledLinkedList and remove that TCB from doubledLinkedList.
Edge cases:
If there is no element in doubledLinked list or given TCB is not in the linkedList, delete function should have no effect.
If there given TCB is head or tail of the doubledLinkedList, removing that TCB will not cause global pointer head and tail lost track of real start and end of doubled linked list.

UNO side
1. Firstly, we send request message from ATMega to UNO and we will print those messages out in UNO's console to verify that whether they are identical.
2. Secondly, we will print health data measurement simulated to UNO's console every time a request message is received. Note that only data explicitly asked by function select message are updated and unrequested data should not change (Health data simulation behaviors are shown in table below).

Meaturement Variable	Behavior on Even time the measurement is requested	Behavior on Odd time measurement is requested

Temperature (exceeding 50)	-2	+1
Temperature (Falling below 15)	+2	-1
Systolic Pressure (when measuring systolic pressure)	+3	-1
Systolic Pressure (when measuring diastolic pressure)	Stay the same	Stay the same
Diastolic Pressure (when measuring Diastolic pressure)	-2	+1
Diastolic Pressure (when measuring systolic pressure)	Stay the same	Stay the same
Pulse Rate (exceeding 40)	# of KHz supplied by function generator	

* If given measurement is not requested, value of that measurement should not change.

3. Thirdly, we need to send a response message through TX and RX port. We will print those received data on ATMega's console to verify against what we sent.

After we make sure that UNO has the desirable behavior, and communication between UNO and ATMega are good, we can move to ATMega to verify UI interaction and dynamic scheduling.

ATMega side
UI mode
UI mode selection behaviors should follow state table supplied below. We have a global variable indicating which mode we are in (TOP level, menu selection, annunciation). Every time it is changed, we will print out that state in Mega's console and update to corresponding UI. We verify printed mode in console and actual UI displayed are the same.

Current UI State	Event	Next UI State
TOP selection level	Menu tab is tapped	Menu level
	Annunciation tab is tapped	Annunciation level
Menu level	Back tab is tapped	TOP selection level
Annunciation level	Back tab is tapped	TOP selection level

* If none of event is triggered, next UI state should be the same as current UI state.

Measurement selection
There are three squares in Menu level, each corresponds to whether you want to measure blood

pressure, temperature and pulse rate. (Green indicating selected and red indicating unselected)
Next, we need to print out values of those flags to verify that in Menu level, colors of squares correspond with global flags whether we want to make measurement or not.

Warning acknowledgement.
In annunciation mode, there is a green square underneath actual data printed. If there is any red warning display and the green square is tapped, in next display, red warning will become orange warning. If warning is still high for five other measurement red warning will reappear.
We should set pressure to out of range and tap acknowledge square and observe whether warning will change color and whether warning color will change back to red after five high measurement.
Edge case: Warning should disappear and acknowledge flag should be cleared, if health data returned back to normal range.

Dynamic scheduler
We are printing number of TCBs in task queue every time we finish scheduling. #number of TCB should match what I described below.

UI Mode	# of TCB
Menu/Top level	0
Annunciation	1 when we are only issuing warning 5 when we need to do all other tasks in the same time

Test Cases
DoubledLinkedList
Case1 :
Empty list, Remove head/tail
Case2 :
Empty list, insert a TCB A
Case3:
A list containing only a TCB
delete(head)
Case4:
A list containing A->B->C->D
Delete(B), insert(E)
Case5:
A list containing A->B->C->D
Delete A and delete D
UNO side
Value of health data to {temperature, systolic, diastolic, pulse rate}
Value of function selction {temp,blood pressure,pulse rate}

Measurement
Case1:
Function genreator frequency 1 kHz
Function select {1, 1, 1}
Health data{75, 80, 80, _}
Case2:
Function genreator frequency 20 kHz
Function select {0, 1, 1}
Health data{75, 80, 80, _}
Case3:
Function genreator frequency 3 kHz
Function select {0, 0, 1}
Health data{60, 70, 80, _}
Case4:
Function genreator frequency 3 kHz
Function select {0, 0, 0}
Health data{75, 80, 80, _}

Communication:
Case1:
UNO send
{9, 80, 80, 80, 80, 0}
Case 2:
UNO send
{9, 0, 0 , 45, 0}
Case 3:
UNO send
{9, 67, 3, 555, 0}
Case 4:
Mega send {start, functionSelect, end}
{9, 0b111, 0}
Case 5:
Mega send {start, functionSelect, end}
{9, 0b101, 0}
Case 6:
Mega send {start, functionSelect, end}
{9, 0b000, 0}

ATMega
Case1:
In top level, tap menu tab.
In top level, tap annunciation tab.
In top level, tap somewhere else
Case 2:

In menu level, tap temperature tab
In menu level, tap blood pressure tab
In menu level, tap pulse rate tab
In menu level, observe that # of TCB should be 0.
In menu level, tap back tab and return back to check whether measurement selection are the

Case3:
In annunciation level, observe data displayed are the same as we set in menu level
In annunciation level, observe # of TCB should be 1 when we only need warning and 5 when all tasks are needed.
In annunciation level, observe value of acknowledge flag when I tap acknowledge tap.
In annunciation level, observe value of acknowledge flag when warnings are kept up.

Test Results

DoubledLinkedList:
Case 1:
List: {}
Case 2:
List: {A}
Case 3:
List; {}
Case 4:
After deleting B
List: {A->C->D}
After inserting E
List:{A->C->D->E}
Case 5:
After deleting A
List{B->C->D}
After deleting D
List{B->C}
Edge cases and normal cases are handled properly

Communication

Case Number	Received data on other side
1	{9, 80, 80, 80, 80, 0}
2	{9, 0, 0 , 45, 0}
3	{9, 67, 3, 555, 0}
4	{9, 0b111, 0}

5	{9, 0b101, 0}
6	{9, 0b000, 0}

Received data are consistent with sent data.

UNO side

Case Number	Provided hz	Computed Pulse Rate
1	1k	123
2	2k	230
3	3k	310
4	4k	450
5	5k	533
6	6k	590

* Test result for other measurements are identical to those in lab2, thus omitted. Please refer lab2 report for more information.

Computed Pulse rate value has a linear relationship with provided hz.

ATMega:
In top level:

Event	UI	Mode printed on console
Menu tab tapped	Change to Menu UI	“MENU”
Annunciation tab tapped	Change to Annunciation UI	“ANNUN”
Somewhere else tapped	Top level UI	“”

In menu level:

Event	UI	# of TCB	Value of Flags printed
Temp tab tapped	Temp tab becomes red	0	0b011

Blood pressure tab tapped	Blood pressure tab becomes red	0	0b001
Pulse rate tab tapped	Pulse rate tab becomes red	0	0b000
Back tab tapped	Top level UI	0	Remain the same

* Each time we make a selection, we exit then return to menu mode, measurement selection remains the same.

In annunciation level:

Event	Value of acknowledge flag
Acknowledge tab tapped	5
Warnings are up for 2 more measurement	3
Warnings are up for 5 more measurement	0
Warning disappear after acknowledge tab tapped	0

* If acknowledge flag is greater than 0, we observed that corresponding warnings become orange and # of TCB is 5 when all tasks are needed and 1 when only warning is needed.

7. Summary and Conclusion

Summary

Lab3 is the second part of a large medical project, blood pressure measurement, where our peripheral system can measure human body temperature, blood pressure and pulse rate, calculates them and translate them into systolic pressure and diastolic pressure. Peripheral system sends data to through communication channel to System Control to calculate, store, and display on display panel. Our users can directly control everything through Display panel to choose to measure whatever they want and highlight(using yellow) the information that are out of normal range. We start this project by modeling the system and using UML to draw block diagram, control flow, data flow and operation. Then we use Uno connecting with pulse generator to measure data in our system and simulate the human body environment on ATMega2560. We first use Code Block IDE to write our prototype where we can see whether our basic codes can execute and have basic behavior. Then we integrate our C code on Arduino IDE and fulfill all the application requirement and display configuration. Finally, for some small bugs like computation error, we put our C code back to Code Block and test it. Then we get our system where for now it can only display a constant data model on our TFT screen/display panel.

1. Timing Issue

Due to the refresh rate of Screen on Uno, the minimum user friendly output's rate should be larger than 10ms.

When choose measurement items, user should click the button and release the button as soon as possible, otherwise it will not change the previous status of measurement.

When choose annunciation, the whole system will stall for 2 seconds due to the process time which is inevitable.

2. Potential Bug.

Continuing press on button without release will make system stall until users release.

3.Test Issue

It is hard to read time in second with small error range. Because our delay can be accurate in 10x ms, so we need to test our output delay is correct or acceptable.

4. Tradeoff

There is no critical tradeoff.

Conclusion

After the lab2, we start to implement a large project, so we have too many things to consider, using more devices to measure data where we need only use one central kernel to control everything, communications and improving static schedule to dynamics schedule which can make system more resource efficiency and fast. In this lab, we first time experience of difficulty of integrate multiple devices , co-work of multiple MCU and communications between MCU. We all know how to program as the beginning of the college but we do not know how to design, so it is worth to take time on using UML drawing data and control flow, and decompose system into piece of module and functions.

8. Individual Contribution

Name	Contribution
Gaohong Liu, Shawn Xu, Yongqin Wang	Software design & testing
Gaohong Liu, Shawn Xu, Yongqin Wang	System integration
Gaohong Liu, Shawn Xu, Yongqin Wang	Lab Report

table 14

9. Appendices

User Manual

After system is ready to work, users now have two thing to choose

1) Measurement Menu
2) Annunciation

Inside Measurement Menu, Users have three things to choose, left button is body temperature, middle button is pulse rate, and right button is blood pressure. If they are in green, it means users are currently measuring and they are in red, it means they are disabled. After finish measurement selection, users could press return button to back to mean menu.

If users press Annunciation button, the system will begin to measure and calculate and display panel will display measured data on panel. User still could use return button to return back to main menu

Source Code

Source code locates at ee474_proj3.zip

10. Reference

[1]Class.ee.washington.edu. (2018). ***EE474 Workload and Grading*. [online] Available at: https://class.ee.washington.edu/474/peckol/admin/expectations.html [Accessed 6 Apr. 2018].**

[2]Hannaford, B., Ecker, A., Emerson, B., Nguyen, A., Lee, G., Aasheim, J. and Cornelius, T.. Peckol, J. (2018). ***EE 474 Project 2 Learning the Development Environment – The Next Step* [ebook] James Peckol, p.18. Available at: https://class.ee.washington.edu/474/peckol/assignments/lab2/project2-Spring18.pdf [Accessed Apr 19, 2018].**