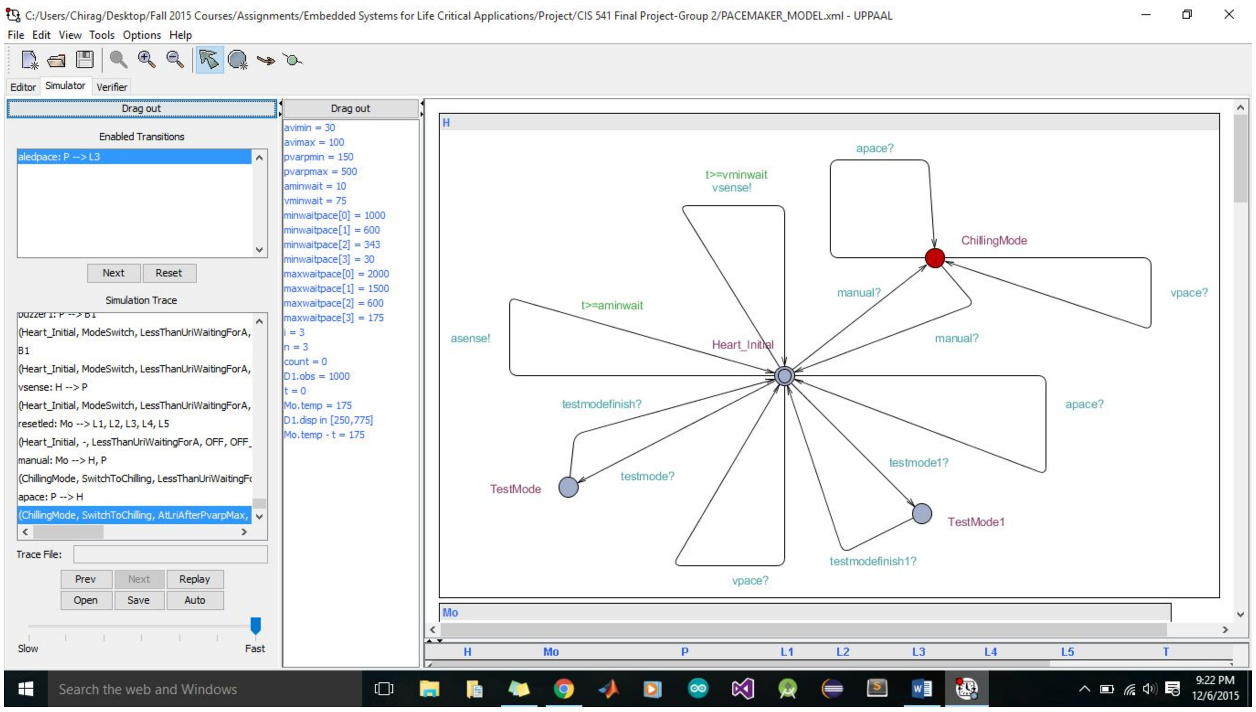
The Pacemaker Challenge

CIS 541 Group 2:
Anupama Kumar anupamak@seas.upenn.edu
Akshay Sriraman akss@seas.upenn.edu
Chirag Shah chirags@seas.upenn.edu
Shruthi Ashok Kumar ashruthi@seas.upenn.edu

A look at our models:



```
aledpace: P --> L3
```

```
buzzer1: P --> B1
(Heart_Initial, ModeSwitch, LessThanUriWaitingForA,
B1
(Heart_Initial, ModeSwitch, LessThanUriWaitingForA,
vsense: H --> P
(Heart_Initial, ModeSwitch, LessThanUriWaitingForA,
resetled: Mo --> L1, L2, L3, L4, L5
(Heart_Initial, -, LessThanUriWaitingForA, OFF, OFF_
manual: Mo --> H, P
(ChillingMode, SwitchToChilling, LessThanUriWaitingF
apace: P --> H
(ChillingMode, SwitchToChilling, AtLriAfterPvarpMax,
```

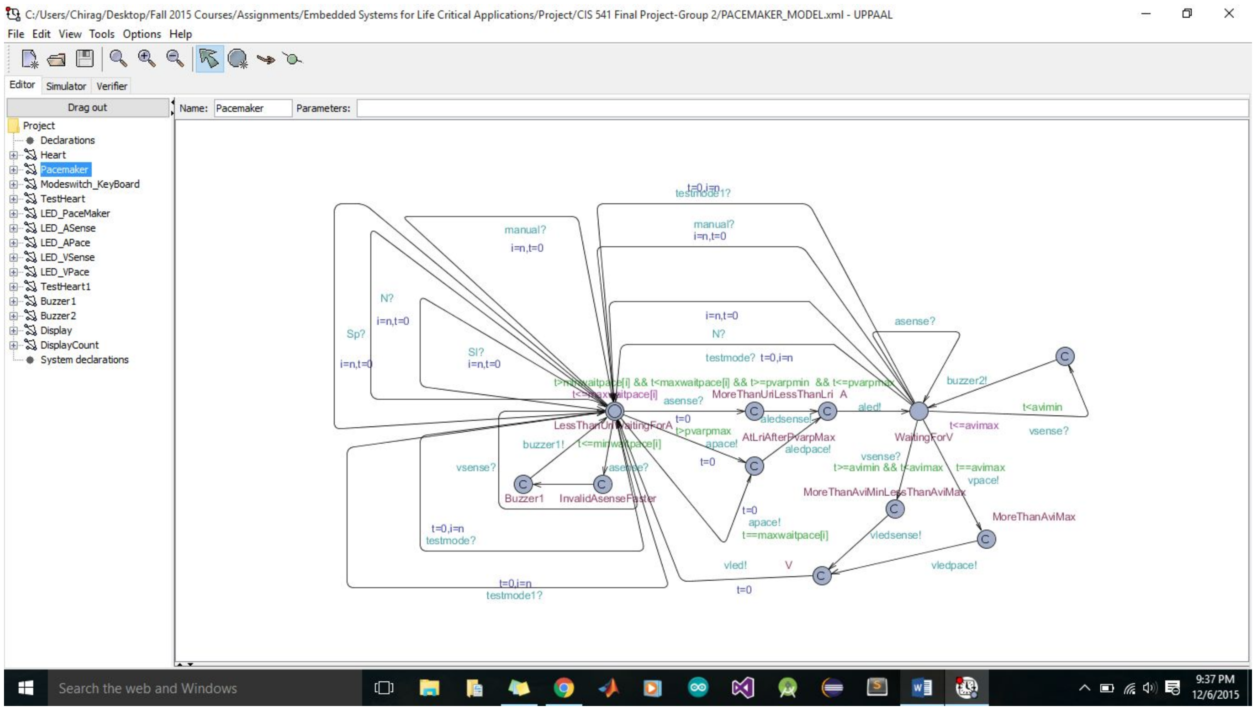
```
avimin = 30
avimax = 100
pvarpmin = 150
pvarpmax = 500
aminwait = 10
vminwait = 75
minwaitpace[0] = 1000
minwaitpace[1] = 600
minwaitpace[2] = 343
minwaitpace[3] = 30
maxwaitpace[0] = 2000
maxwaitpace[1] = 1500
maxwaitpace[2] = 600
maxwaitpace[3] = 175
i = 3
n = 3
count = 0
D1.obs = 1000
t = 0
Mo.temp = 175
D1.disp in [250,775]
Mo.temp - t = 175
```

Heart Model: This is a screenshot for UPPAAL implementation of a random heart. The random heart generates Asignals and Vsignals at random intervals and is modeled in such a way that it may work correctly under some situations while in some situations it may not.

Pacemaker Model:
Pacemaker model is shown in the screenshot shown below. The pacemaker is modeled in such a way that it has various modes and it keeps pacing the heart at a constant heart rate only when needed. It also has functionalities to reject faulty heart beats. There are various signals from the pacemaker which control the LED, Display and the alarm signals. These together constitute of our pacemaker model in UPPAAL.
The hard timing specifications taken into consideration for the Pacemaker model are as follows: AVI, LRI, URI, PVARP and VRP. These are the 5 timings which are taken care of in our models. Amongst these, URI and LRI are different for different modes. The modes in our model are Sleep, Normal, Sports and Manual.

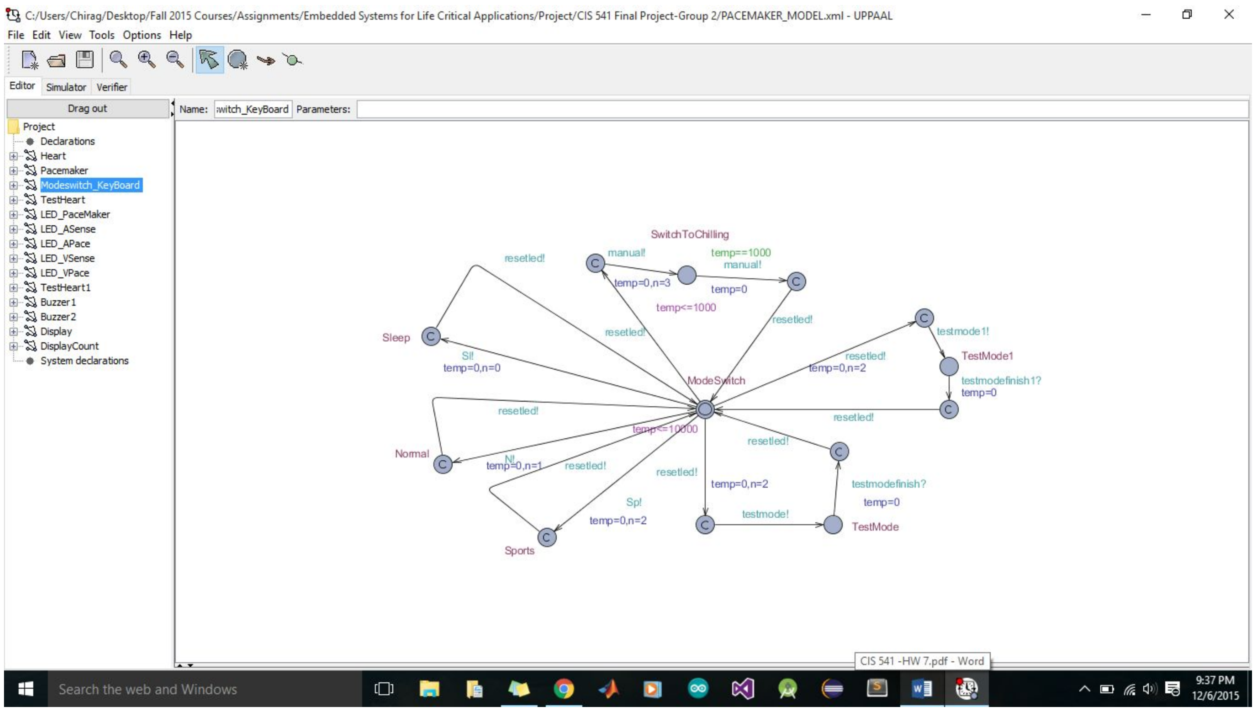
Along with modes and timing constraints, the UPPAAL model also has a simulated keyboard which will switch states as and when needed. This is done by creating a mode switch keyboard on UPPAAL and then syncing it with the pacemaker.

Pacemaker's Screen Shot:

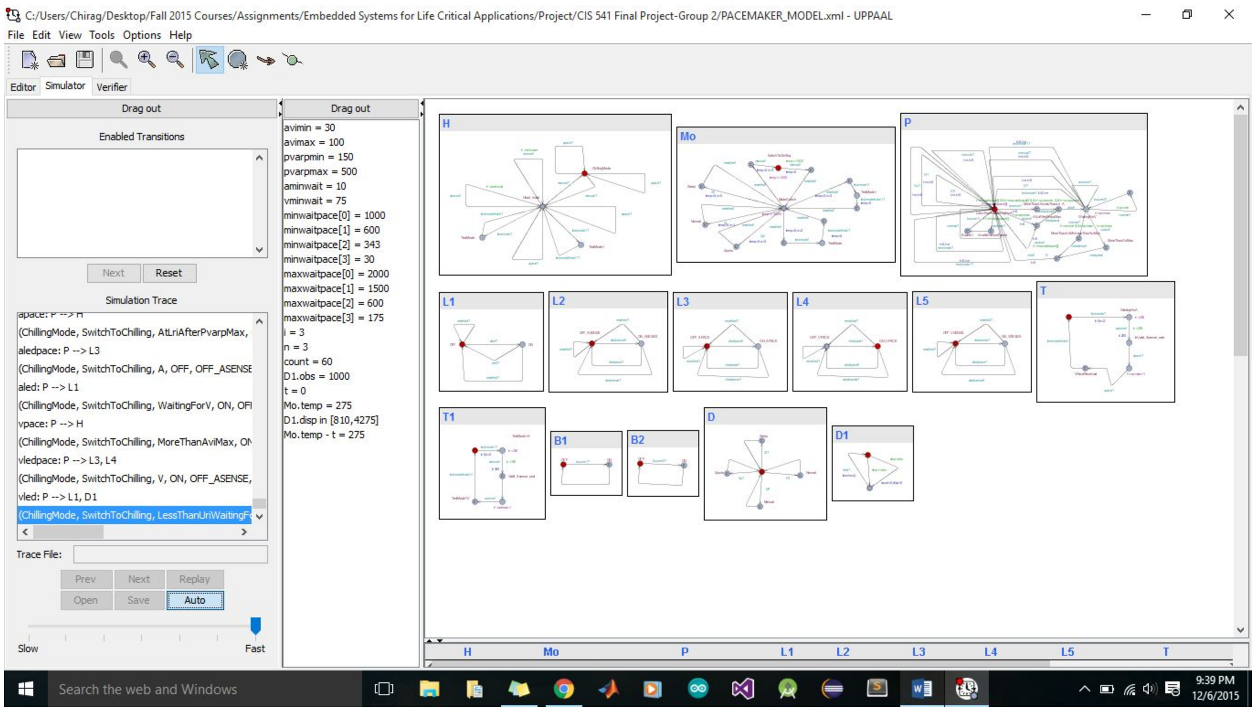


2

Mode Switch Keyboard ScreenShot:



The **whole system in UPPAAL** can be viewed in the next screenshot:


C:/Users/Chirag/Desktop/Fall 2015 Courses/Assignments/Embedded Systems for Life Critical Applications/Project/CIS 541 Final Project-Group 2/PACEMAKER_MODEL.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
Drag out
Enabled Transitions
Next Reset
Simulation Trace
Drag out
H
Mo
P
L1
L2
L3
L4
L5
T
T1
B1
B2
D
D1
Trace File:
Prev Next Replay
Open Save Auto
Slow Fast
Search the web and Windows
9:39 PM
12/6/2015

```
apace: P --> H
(ChillingMode, SwitchToChilling, AtLriAfterPvarpMax,
aledpace: P --> L3
(ChillingMode, SwitchToChilling, A, OFF, OFF_ASENSE
aled: P --> L1
(ChillingMode, SwitchToChilling, WaitingForV, ON, OFI
vpace: P --> H
(ChillingMode, SwitchToChilling, MoreThanAviMax, ON
vledpace: P --> L3, L4
(ChillingMode, SwitchToChilling, V, ON, OFF_ASENSE,
vled: P --> L1, D1
(ChillingMode, SwitchToChilling, LessThanUriWaitingF
```

```
avimin = 30
avimax = 100
pvarpmin = 150
pvarpmax = 500
aminwait = 10
vminwait = 75
minwaitpace[0] = 1000
minwaitpace[1] = 600
minwaitpace[2] = 343
minwaitpace[3] = 30
maxwaitpace[0] = 2000
maxwaitpace[1] = 1500
maxwaitpace[2] = 600
maxwaitpace[3] = 175
i = 3
n = 3
count = 60
D1.obs = 1000
t = 0
Mo.temp = 275
D1.disp in [810,4275]
Mo.temp - t = 275
```

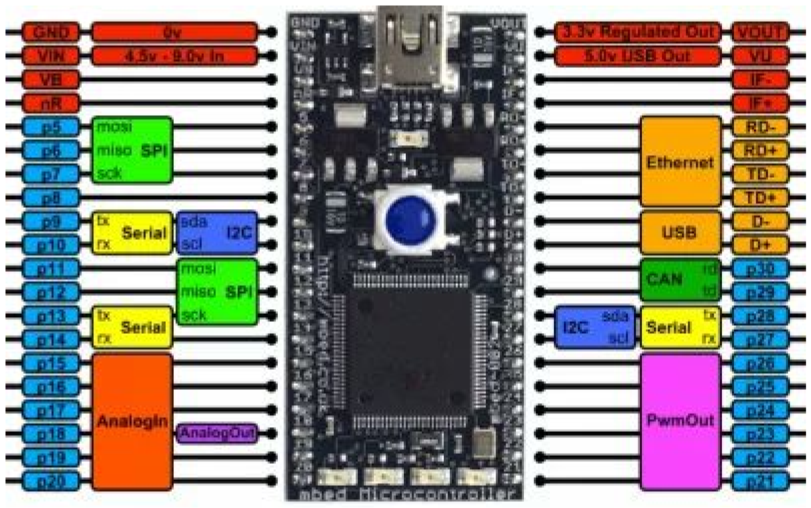
There are a total of 29 queries we have verified in our pacemaker model. Some of the **key queries** are summarized as follows.

1. After a valid ASignal or an Apace has come, the model is going to wait for V (Signal or Pace).
2. After a valid VSginal or a Vpace has come, the model is going to wait for A(Signal or Pace).
3. All the ASignals before PVARP and before URI are going to be rejected because these Atrial pulses are not bounded by the limit on the timing constraints.
4. The model is deadlock free.
5. A valid ASignal can only come after PVARP and URI has elapsed where both the timings are measured with references to different states. An Apace can come only after LRI and pVARP where both the timings are measured with different reference states.

All other queries can be successfully verified in the UPPAAL model. The queries also verify the properties of the display and the test cases and hence verify the correctness of our model.

Implementation:

Hardware Platform used: **MBED LPC1768 32-bit ARM® Cortex™-M3**



Software used for Development:

Programming Language: C++

IDE : MBED's online Platform

Link to access the code:

https://github.com/chirags27/Pacemaker-Challenge

Link to Video:

https://www.youtube.com/watch?v=qz2C5U5_JB4

Explanation of overall structure of the code:
For the **Pacemaker tester**, there are threads for sending Asignals and Vsignal and accepting Apace and Vpace. There is a separate thread which puts the pacemaker tester in the test mode as well. The pacemaker is tested rigorously in this mode.
For the initial 4 threads:
Asignal: This signal is sent randomly after a small aminwait period.
Vsignal: This signal is sent randomly after a small vminwait period.
Apace: This is used to accept Apaces from the pacemaker when Asignals are not valid.
Vpace: This is used to accept Vpaces from the pacemaker when Vsignals are not valid.

Testmode:
Test 1 : To test a perfect heart.
Test 2 : To test a dysfunctional heart.
Test 3 : To test Asignal is before LRI and after PVARP and Vpace arrives after avimin.
Test 4 : To test Vsignal is after avimin and before Avimax and Apace arrives after LRI and PVARP.
Test 5 : To test the alarm functionalities when the heart is not beating and the pacemaker signals aren't received.

To test the case when Signals aren't sent to the Pacemaker and the Paces are temporarily disconnected from the pacemaker using the software (By stopping the threads for detecting the paces). This causes the buzzer to Buzz.

Also, one more functionality of the heart is the display. The LCD connected to the tester will display the last calculated heart rate.
The different modes are: Random (Default)
Test Mode
Manual Mode
The different functionalities like LCD display were also added to the tester to display the heart rate and current mode of the heart.

Pacemaker:
In this Implementation of DDD mode- pacemaker model we have the Pacemaker and the PM tester Mbeds. The implementation of this model is based on our Uppaal model that we constructed and verified in Milestone 1.

Pacemaker Implementation:
The pacemaker monitors the heart activities on both the atrium and ventricle channels and sends the signals Vpace and Apace to the heart if the heart doesn't beat according to the timing cycles

determined based on the mode in which heart is . Each automata that we modelled in milestone 1 has been implemented as a thread.
The threads are:

1) Thread pacemaker(&pacemakerThread);
2) Thread buzzerThread(&buzzer_);
3) Thread modeSwitch(&modeSwitchFunctionality);
   (For switching between the 3 modes)

The basic timing implementations follow the timing cycles of LRI, VRP, PVARP, and AVI.

The models of the pacemaker are as follows:
1) Manual - where we give ‘a’ for apace and ‘v’ for vpace and ‘e’ for exiting the manual mode.
2) Normal
3) Sleep
4) Exercise

Communication between the PM and the PM tester :
The pacemaker and the PM tester communicate through the 4 I/O pins .
A/V pacing outputs and A/V sensing inputs on the PM,
A/V pacing inputs and A/V sensing outputs on the tester.
(In case of no event the I/O pins output 0)

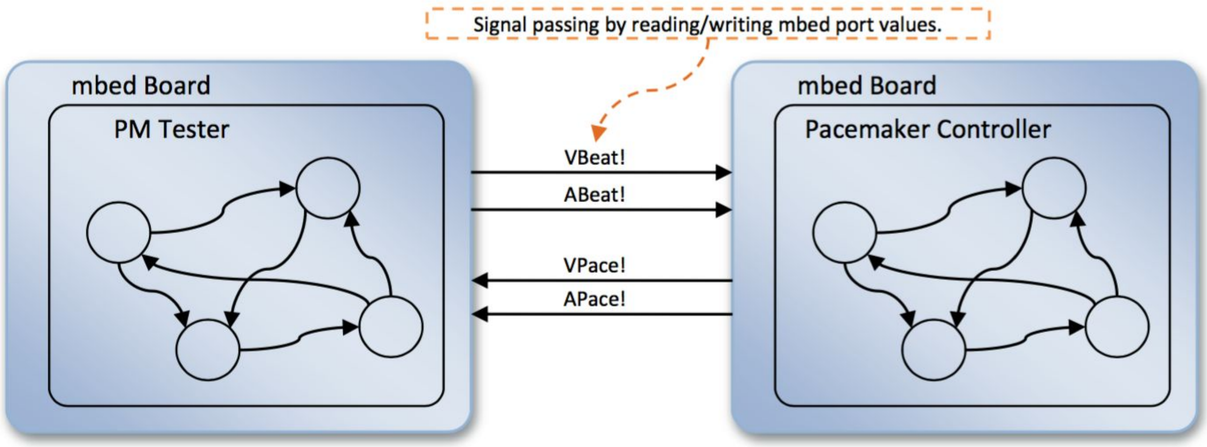


Figure 1 Basic architecture of the pacemaker system

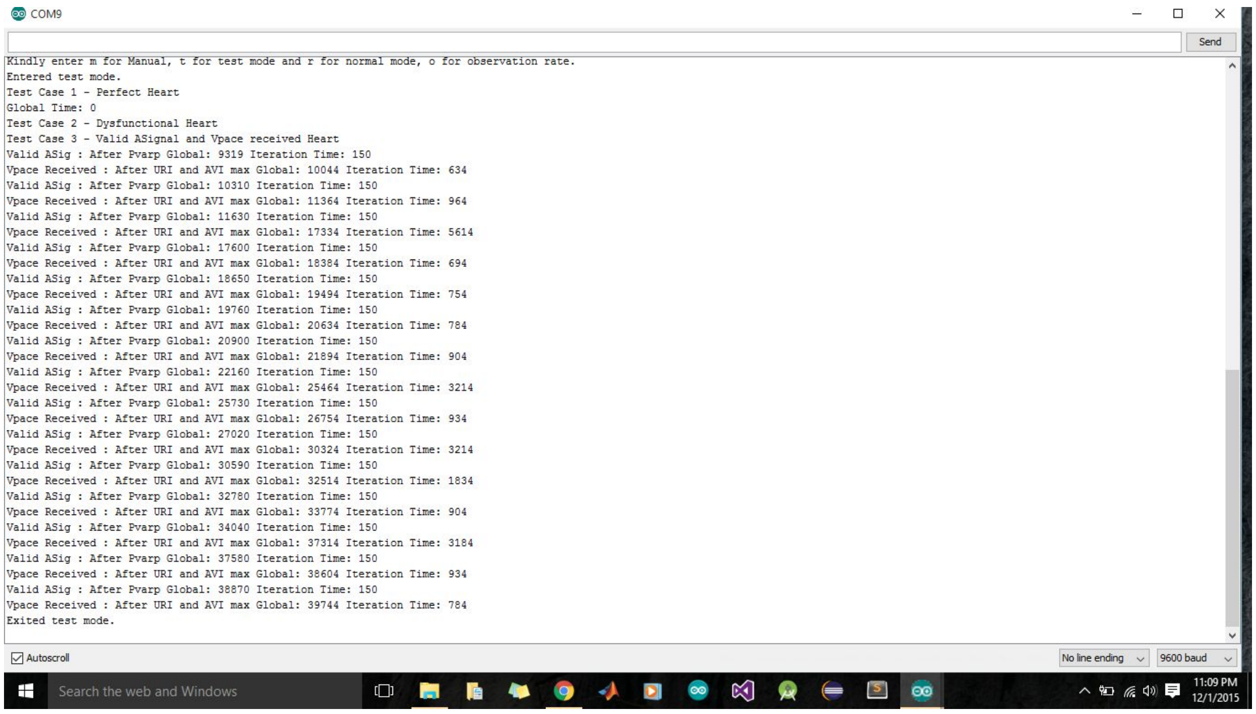
In case of sensing/pacing event, the PM/tester outputs 1 on the corresponding pins of the Mbed for the time period set by observational interval (1-10 ms).

We use interrupts to detect the signal change on the I/O channel .
void InterruptTestingReq() { testing = true; }
The receiver is interrupted to handle this signal and then resumes its execution.

The pacemaker thread initializes with all LEDs at the off state. From the Mbed LEDs we can recognise whether pacing or sensing is output.

Conditions :
For the pacemaker , we wait until the pacemaker outputs a ventricular pace- Vpace and start the test after that.

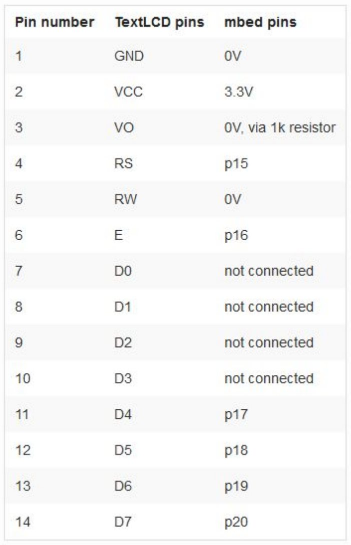
Generating traces :
Each trace contains signals Asense, Vsense, Apace, Vpace of ALL FOUR communication channels, i.e., the print out statements indicate the signals on the four channels (A/V sensing and pacing) on the COM screen during testing.



```
Kindly enter m for Manual, t for test mode and r for normal mode, o for observation rate.
Entered test mode.
Test Case 1 - Perfect Heart
Global Time: 0
Test Case 2 - Dysfunctional Heart
Test Case 3 - Valid ASignal and Vpace received Heart
Valid ASig : After Pvarp Global: 9319 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 10044 Iteration Time: 634
Valid ASig : After Pvarp Global: 10310 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 11364 Iteration Time: 964
Valid ASig : After Pvarp Global: 11630 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 17334 Iteration Time: 5614
Valid ASig : After Pvarp Global: 17600 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 18384 Iteration Time: 694
Valid ASig : After Pvarp Global: 18650 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 19494 Iteration Time: 754
Valid ASig : After Pvarp Global: 19760 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 20634 Iteration Time: 784
Valid ASig : After Pvarp Global: 20900 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 21894 Iteration Time: 904
Valid ASig : After Pvarp Global: 22160 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 25464 Iteration Time: 3214
Valid ASig : After Pvarp Global: 25730 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 26754 Iteration Time: 934
Valid ASig : After Pvarp Global: 27020 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 30324 Iteration Time: 3214
Valid ASig : After Pvarp Global: 30590 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 32514 Iteration Time: 1834
Valid ASig : After Pvarp Global: 32780 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 33774 Iteration Time: 904
Valid ASig : After Pvarp Global: 34040 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 37314 Iteration Time: 3184
Valid ASig : After Pvarp Global: 37580 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 38604 Iteration Time: 934
Valid ASig : After Pvarp Global: 38870 Iteration Time: 150
Vpace Received : After URI and AVI max Global: 39744 Iteration Time: 784
Exited test mode.
```

How to run the code:
The .bin files are generated by compiling, assembling and linking the .cpp code using the online compiler. The .bin files can be uploaded directly into 2 mbed's and the following connections can be made for preparing the implemented model and get it working.

Hardware Connections :
Connections between LCD and pacemaker Mbed :



Pin number	TextLCD pins	mbed pins
1	GND	0V
2	VCC	3.3V
3	VO	0V, via 1k resistor
4	RS	p15
5	RW	0V
6	E	p16
7	D0	not connected
8	D1	not connected
9	D2	not connected
10	D3	not connected
11	D4	p17
12	D5	p18
13	D6	p19
14	D7	p20

The LCD is connected to the mbed that represents the pacemaker using the pin connections as shown above. LCD is connected to the heart in similar fashion.

Connections and pacemaker Mbed and LEDs :

There are two sets of LEDs - One represents the pacing signals being given to the heart. The other glows whenever there is an Asignal and Vsignal received from the heart mbed.

Pin 12 is connected to a red LED that represents Apace. It glows whenever this signal is generated.

Pin 13 is connected to a green LED that represents Vpace.It glows whenever this signal is generated.

Pin 10 is connected to a red LED that represents Asignal.It glows whenever this signal is received.

Pin 11 is connected to a green LED that represents Vsignal.It glows whenever this signal is received.

Connections between heart Mbed and pacemaker Mbed:

Pin 5 (DigitalOut) of heart mbed is connected to pin 5(Digitaln) of pacemaker mbed. This connection represents Asignal.

Pin 6(DigitalOut) of heart mbed is connected to pin 6(DigitalIn) of pacemaker mbed.This connection represents Vsignal.

Pin 7(Digitaln) of heart mbed is connected to pin 7(DigitalOut) of pacemaker mbed. This connection represents Apace.

Pin 8(DigitalIn) of heart mbed is connected to pin 8(DigitalOut) of pacemaker mbed. This connection represents Vpace.

Pin 21 of heart mbed is connected to pin 21(DigitalOut) of pacemaker mbed for synchronization during the functioning of test mode. This is used to receive and send a ready signal.

Pin 22 of heart mbed is connected to pin 22 of pacemaker mbed. This sends and receives an acknowledgement signal.

Pin 23 of heart mbed is connected to pin 23 of pacemaker mbed. This is to model the feedback from the SA Node to the pacemaker as an acknowledgement of a valid/useful Apace.

Pin 24 of heart mbed is connected to pin 24 of pacemaker mbed. This is to model the feedback from the VA Node to the pacemaker as an acknowledgement of a valid/useful Vpace.

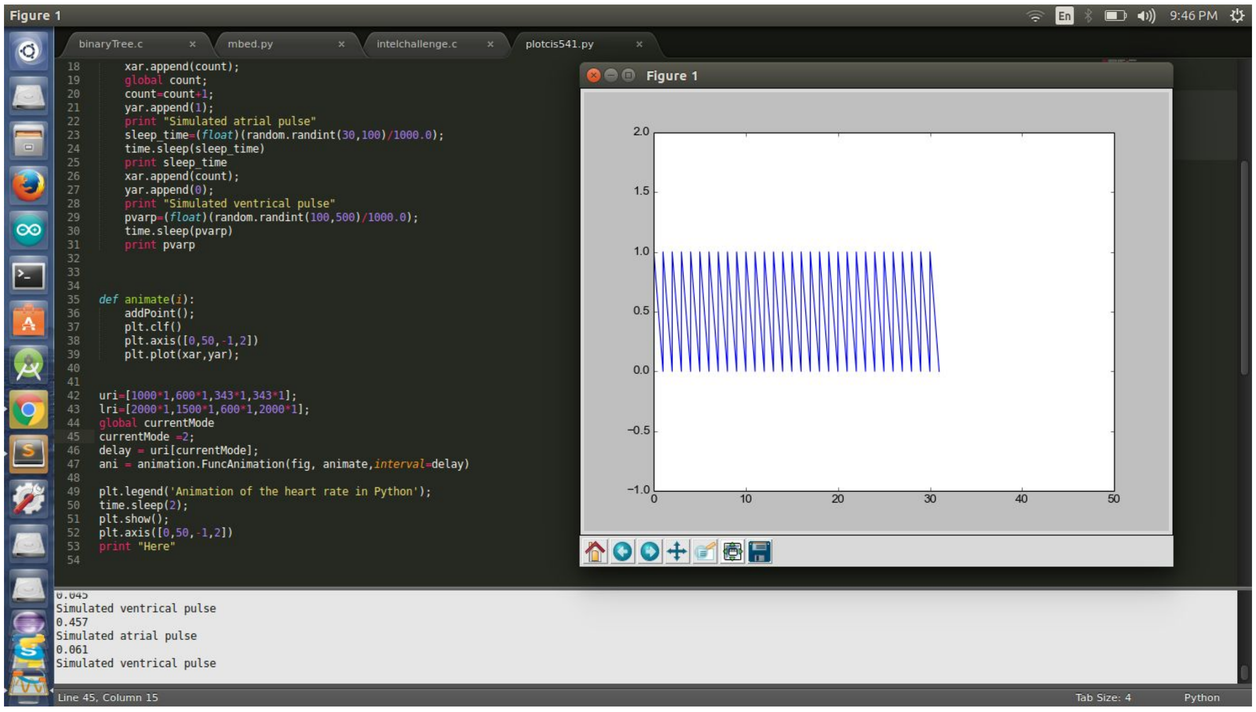
Connections between pacemaker Mbed and buzzer:

Pin 25 of the pacemaker mbed is connected to the buzzer, which is used to model a failure of the pacemaker.

Connections between pacemaker Mbed and 3D heart :

Pins 27 and 28 of the pacemaker mbed is connected to the 3D Heart, which is used to model an assisted (properly functioning) dysfunctional heart.

Python Plots for Heart rate:



```python
    xar.append(count);
    global count;
    count=count+1;
    yar.append(1);
    print "Simulated atrial pulse"
    sleep_time=(float)(random.randint(30,100)/1000.0);
    time.sleep(sleep_time)
    print sleep_time
    xar.append(count);
    yar.append(0);
    print "Simulated ventrical pulse"
    pvarp=(float)(random.randint(100,500)/1000.0);
    time.sleep(pvarp)
    print pvarp



def animate(i):
    addPoint();
    plt.clf()
    plt.axis([0,50,-1,2])
    plt.plot(xar,yar);


uri=[1000*1,600*1,343*1,343*1];
lri=[2000*1,1500*1,600*1,2000*1];
global currentMode
currentMode =2;
delay = uri[currentMode];
ani = animation.FuncAnimation(fig, animate,interval=delay)

plt.legend('Animation of the heart rate in Python');
time.sleep(2);
plt.show();
plt.axis([0,50,-1,2])
print "Here"
```

```
0.045
Simulated ventrical pulse
0.457
Simulated atrial pulse
0.061
Simulated ventrical pulse
```

```
Line 45, Column 15
```

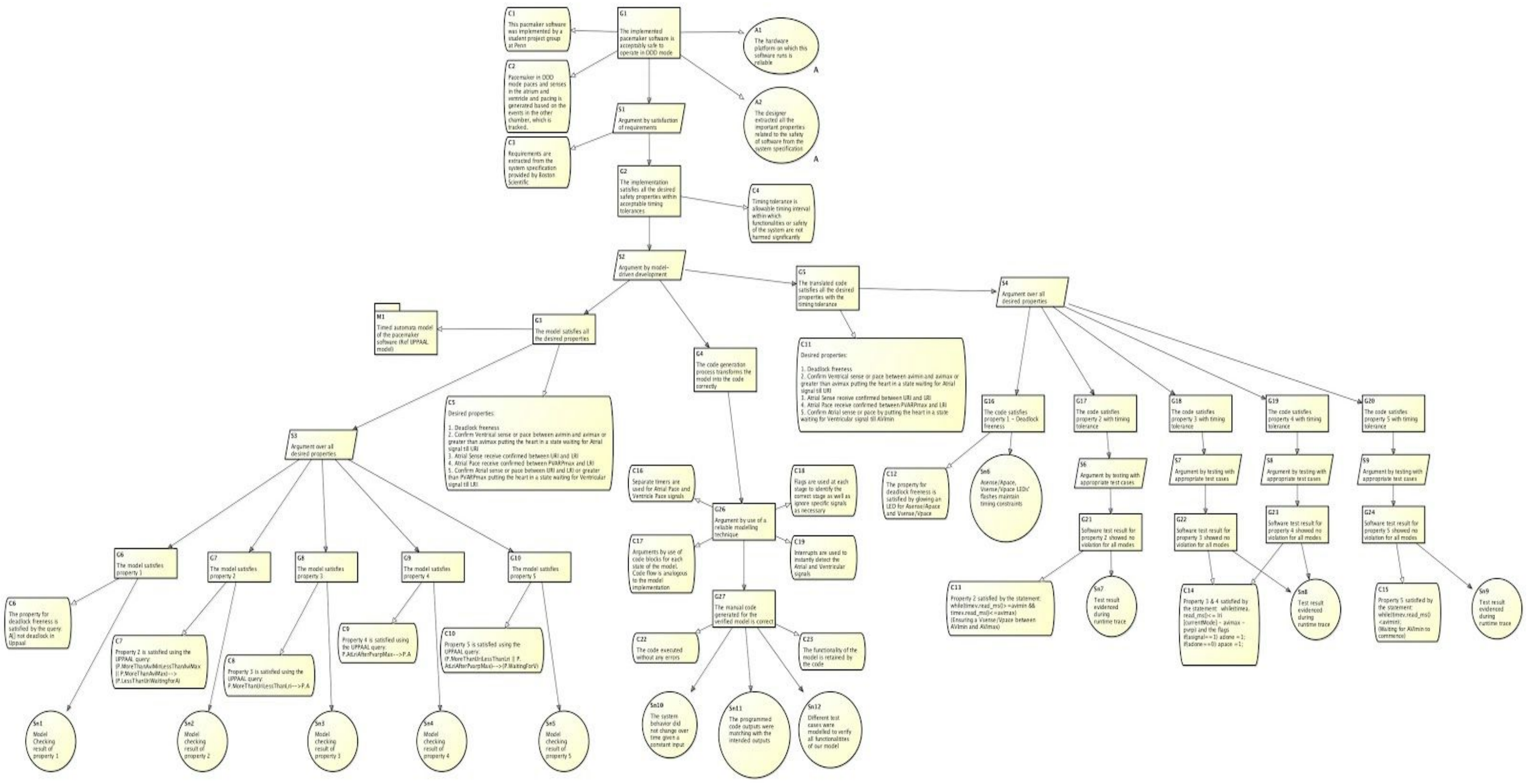
Analysis of experimental results:

ASignal before URI - pvarp - avimax is rejected. This is because if not, the heart rate will exceed the given limit for a particular mode. After a Vpace or a Vsignal, the time of Pvarp and URI - pvarp - avimax is allowed to elapse and then Asignal is allowed to come until LRI- Pvarp - Avimax. This ensures that if any Asignal doesn't come in this interval, the pacemaker will send an Apace. This ensures proper operation of the heart. After a Asignal or Apace has come, a period of avimin is elapsed and then Vsignal is checked unitl Avimax. If no valid VSginal comes, Vpace is given and then the VARP and PVARP signals begin and then the cycle repeats itself. This takes place in a deterministic manner.

Assurance Cases :

Assurance is a justified measure of confidence that a system will function as intended in its environment of use.

Assurance case (complete):


G1
The implemented pacemaker software is acceptably safe to operate in DDD mode
C1
This pacemaker software was implemented by a student project group at Penn
C2
Pacemaker in DDD mode paces and senses in the atrium and ventricle and pacing is generated based on the events in the other chamber, which is tracked.
A1
The hardware platform on which this software runs is reliable
A
A2
The designer extracted all the important properties related to the safety of software from the system specification
A
S1
Argument by satisfaction of requirements
C3
Requirements are extracted from the system specification provided by Boston Scientific
G2
The implementation satisfies all the desired safety properties within acceptable timing tolerances
C4
Timing tolerance is allowable timing interval within which functionalities or safety of the system are not harmed significantly
S2
Argument by model-driven development
G5
The translated code satisfies all the desired properties with the timing tolerance
S4
Argument over all desired properties
M1
Timed automata model of the pacemaker software (Ref UPPAAL model)
G3
The model satisfies all the desired properties
G4
The code generation process transforms the model into the code correctly
C11
Desired properties:
1. Deadlock freeness
2. Confirm Ventrical sense or pace between avimin and avimax or greater than avimax putting the heart in a state waiting for Atrial signal till URI
3. Atrial Sense receive confirmed between URI and LRI
4. Atrial Pace receive confirmed between PVARPmax and LRI
5. Confirm Atrial sense or pace by putting the heart in a state waiting for Ventricular signal till AVImin
C5
Desired properties:
1. Deadlock freeness
2. Confirm Ventrical sense or pace between avimin and avimax or greater than avimax putting the heart in a state waiting for Atrial signal till URI
3. Atrial Sense receive confirmed between URI and LRI
4. Atrial Pace receive confirmed between PVARPmax and LRI
5. Confirm Atrial sense or pace between URI and LRI or greater than PVARPmax putting the heart in a state waiting for Ventricular signal till LRI
S3
Argument over all desired properties
G6
The model satisfies property 1
G7
The model satisfies property 2
G8
The model satisfies property 3
G9
The model satisfies property 4
G10
The model satisfies property 5
C6
The property for deadlock freeness is satisfied by the query: A[] not deadlock in Uppaal
C7
Property 2 is satisfied using the UPPAAL query: (P.MoreThanAviMinLessThanAviMax || P.MoreThanAviMax)--> (P.LessThanUriWaitingforA)
C8
Property 3 is satisfied using the UPPAAL query: P.MoreThanUriLessThanLri-->P.A
C9
Property 4 is satisfied using the UPPAAL query: P.AtLriAfterPvarpMax-->P.A
C10
Property 5 is satisfied using the UPPAAL query: (P.MoreThanUriLessThanLri || P.AtLriAfterPvarpMax)-->(P.WaitingForV)
Sn1
Model Checking result of property 1
Sn2
Model checking result of property 2
Sn3
Model checking result of property 3
Sn4
Model checking result of property 4
Sn5
Model checking result of property 5
C16
Separate timers are used for Atrial Pace and Ventricle Pace signals
G26
Argument by use of a reliable modelling technique
C18
Flags are used at each stage to identify the correct stage as well as ignore specific signals as necessary
C17
Arguments by use of code blocks for each state of the model. Code flow is analogous to the model implementation
C19
Interrupts are used to instantly detect the Atrial and Ventricular signals
G27
The manual code generated for the verified model is correct
C22
The code executed without any errors
C23
The functionality of the model is retained by the code
Sn10
The system behavior did not change over time given a constant input
Sn11
The programmed code outputs were matching with the intended outputs
Sn12
Different test cases were modelled to verify all functionalities of our model
G16
The code satisfies property 1 - Deadlock freeness
G17
The code satisfies property 2 with timing tolerance
G18
The code satisfies property 3 with timing tolerance
G19
The code satisfies property 4 with timing tolerance
G20
The code satisfies property 5 with timing tolerance
C12
The property for deadlock freeness is satisfied by glowing an LED for Asense/Apace and Vsense/Vpace
Sn6
Asense/Apace, Vsense/Vpace LEDs' flashes maintain timing constraints
S6
Argument by testing with appropriate test cases
S7
Argument by testing with appropriate test cases
S8
Argument by testing with appropriate test cases
S9
Argument by testing with appropriate test cases
G21
Software test result for property 2 showed no violation for all modes
G22
Software test result for property 3 showed no violation for all modes
G23
Software test result for property 4 showed no violation for all modes
G24
Software test result for property 5 showed no violation for all modes
C13
Property 2 satisfied by the statement: while(timev.read_ms()>=avimin && timev.read_ms()<=avimax) (Ensuring a Vsense/Vpace between AVImin and AVImax)
Sn7
Test result evidenced during runtime trace
C14
Property 3 & 4 satisfied by the statement: while(timea.read_ms()<= lri [currentMode] - avimax - pvrp) and the flags if(asignal==1) adone = 1; if(adone==0) apace =1;
Sn8
Test result evidenced during runtime trace
C15
Property 5 satisfied by the statement: while(timev.read_ms() <avimin); (Waiting for AVImin to commence)
Sn9
Test result evidenced during runtime trace

The overall claim of the project is that the implemented pacemaker software is acceptably safe to operate in DDD mode. Also the context of these claims are the following:

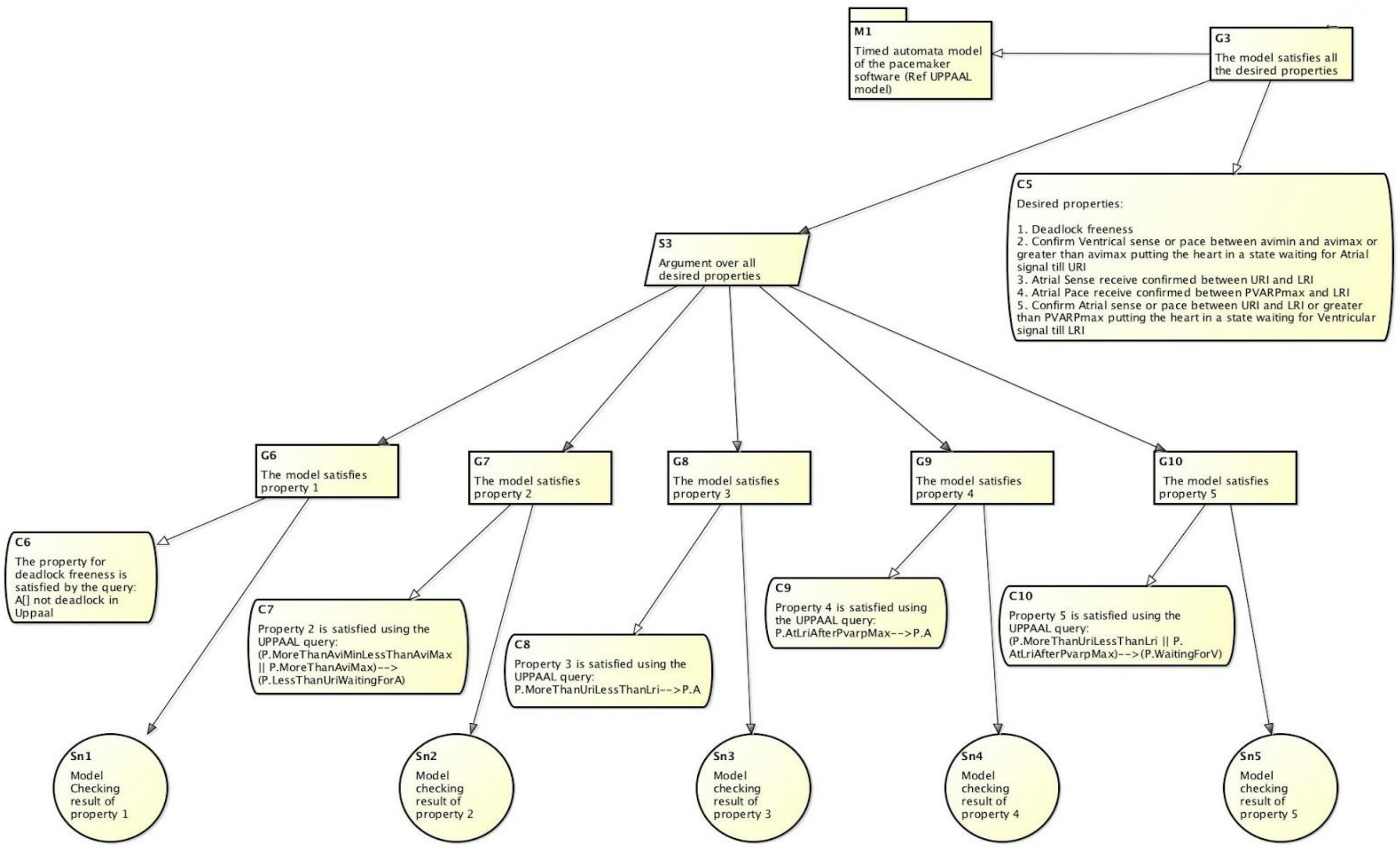
★ Pacemaker in DDD mode paces and senses in both atrial and ventricle and pacing is inhibited when the pacemaker gets a sensing signal.

★ Requirements are extracted from the system specification provided by Boston Scientific.

★ Timing tolerance is allowable timing interval within which functionalities or safety of the system are not harmed significantly.

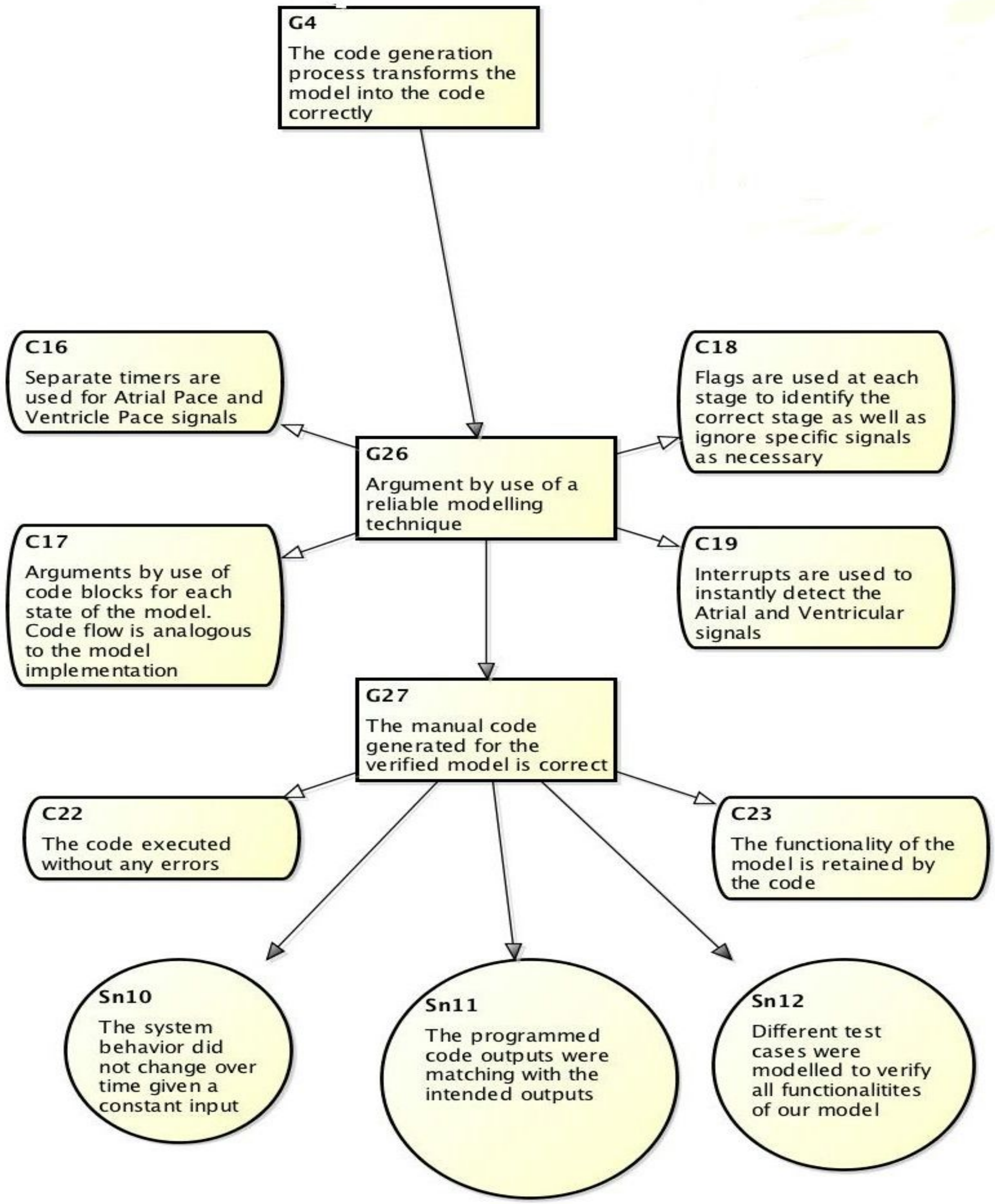
These claims are also made under the assumption that the hardware platform on which this software runs is reliable and the designer extracted all the important properties related to the software safety from the system specification. Further there are 3 sub goals based on code generation , timing tolerance and modelling claims.

Assurance Cases (Part 1):


M1
Timed automata model of the pacemaker software (Ref UPPAAL model)
G3
The model satisfies all the desired properties
C5
Desired properties:
1. Deadlock freeness
2. Confirm Ventrical sense or pace between avimin and avimax or greater than avimax putting the heart in a state waiting for Atrial signal till URI
3. Atrial Sense receive confirmed between URI and LRI
4. Atrial Pace receive confirmed between PVARPmax and LRI
5. Confirm Atrial sense or pace between URI and LRI or greater than PVARPmax putting the heart in a state waiting for Ventricular signal till LRI
S3
Argument over all desired properties
G6
The model satisfies property 1
G7
The model satisfies property 2
G8
The model satisfies property 3
G9
The model satisfies property 4
G10
The model satisfies property 5
C6
The property for deadlock freeness is satisfied by the query: A[] not deadlock in Uppaal
C7
Property 2 is satisfied using the UPPAAL query: (P.MoreThanAviMinLessThanAviMax || P.MoreThanAviMax)--> (P.LessThanUriWaitingForA)
C8
Property 3 is satisfied using the UPPAAL query: P.MoreThanUriLessThanLri-->P.A
C9
Property 4 is satisfied using the UPPAAL query: P.AtLriAfterPvarpMax-->P.A
C10
Property 5 is satisfied using the UPPAAL query: (P.MoreThanUriLessThanLri || P. AtLriAfterPvarpMax)-->(P.WaitingForV)
Sn1
Model Checking result of property 1
Sn2
Model checking result of property 2
Sn3
Model checking result of property 3
Sn4
Model checking result of property 4
Sn5
Model checking result of property 5

The assurance cases for modelling claims are based on the main goal that the model satisfies all the desired properties stated above with reference to the UPPAAL model. We supply an argument over all the properties. This arguments are claimed by providing queries to verify these properties. These queries satisfy the properties mentioned above and their results proving to be evidences for these claims thereby validating the main goal.

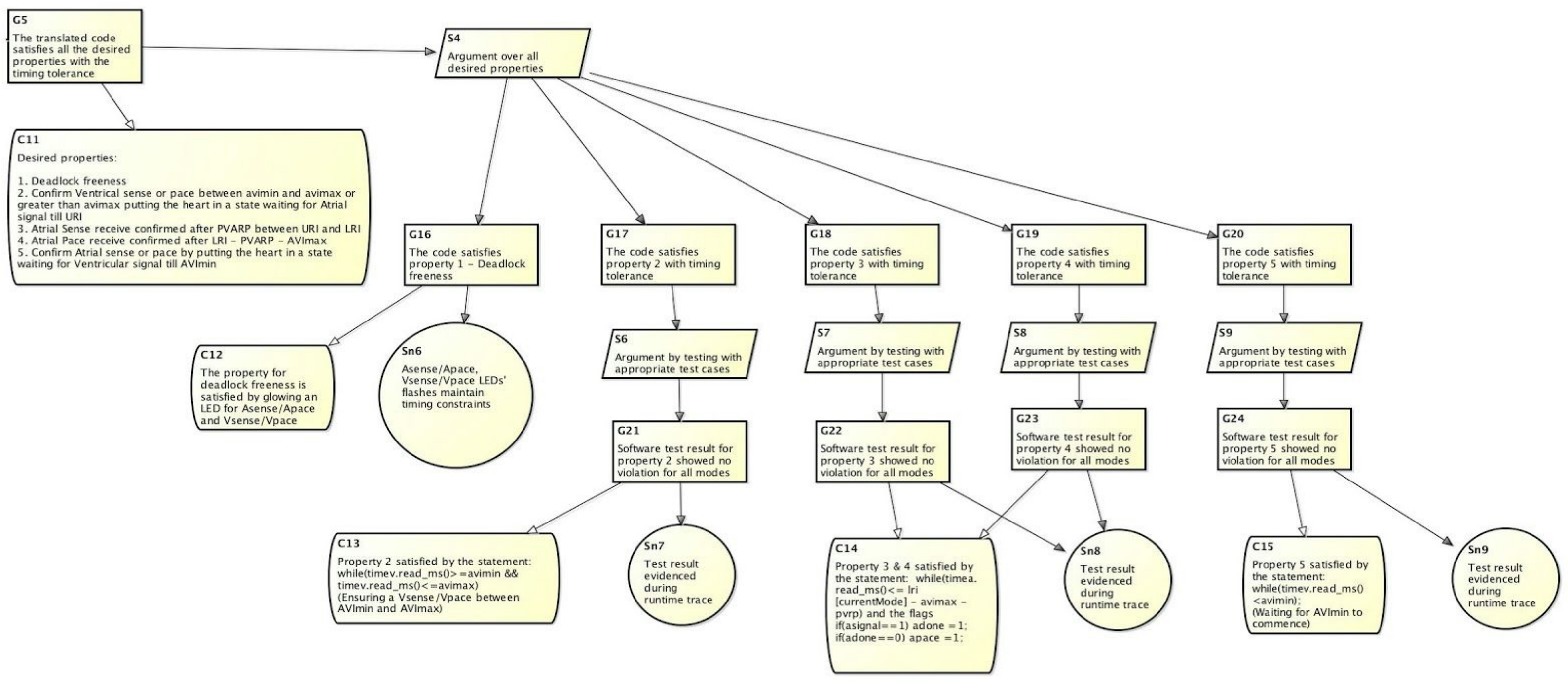
Assurance Cases(Part 2):


G4
The code generation process transforms the model into the code correctly
C16
Separate timers are used for Atrial Pace and Ventricle Pace signals
C18
Flags are used at each stage to identify the correct stage as well as ignore specific signals as necessary
G26
Argument by use of a reliable modelling technique
C17
Arguments by use of code blocks for each state of the model. Code flow is analogous to the model implementation
C19
Interrupts are used to instantly detect the Atrial and Ventricular signals
G27
The manual code generated for the verified model is correct
C22
The code executed without any errors
C23
The functionality of the model is retained by the code
Sn10
The system behavior did not change over time given a constant input
Sn11
The programmed code outputs were matching with the intended outputs
Sn12
Different test cases were modelled to verify all functionalitites of our model

The assurance cases for the code generation are modelled with a main goal that the code generation process transforms the UPPAAL model into the code correctly. We argue this by the use of a logically reliable modeling technique. We provide justification by pointing out the use of two different timers to monitor the Apace and the Vpace signals, to guarantee the accuracy of the interval during which the heart is monitored for signals. Also there are flags used at each stage to identify the current stage,and these flag updates are triggered by timer interrupts. Therefore the whole system is up to date every millisecond, so whichever stage the pacemaker is currently in, the update would be reflected and acted accordingly.

The other goal being that the code synthesis for the verified model is correct. This goal is being supported by claims that the code is executed, without any errors and that the functionality of the UPPAAL model was preserved when transferring to code .These claims can be justified as various test cases were designed and tested on the model. All test results matched the expected results. The precision of the pacemaker system does not vary over time as it was tested for a long duration. The different test cases were modeled to verify different functionalities of our model, and thereby concluding our goal.

Assurance Cases(Part 3):


G5
The translated code satisfies all the desired properties with the timing tolerance
S4
Argument over all desired properties
C11
Desired properties:
1. Deadlock freeness
2. Confirm Ventrical sense or pace between avimin and avimax or greater than avimax putting the heart in a state waiting for Atrial signal till URI
3. Atrial Sense receive confirmed after PVARP between URI and LRI
4. Atrial Pace receive confirmed after LRI - PVARP - AVImax
5. Confirm Atrial sense or pace by putting the heart in a state waiting for Ventricular signal till AVImin
G16
The code satisfies property 1 - Deadlock freeness
G17
The code satisfies property 2 with timing tolerance
G18
The code satisfies property 3 with timing tolerance
G19
The code satisfies property 4 with timing tolerance
G20
The code satisfies property 5 with timing tolerance
C12
The property for deadlock freeness is satisfied by glowing an LED for Asense/Apace and Vsense/Vpace
Sn6
Asense/Apace, Vsense/Vpace LEDs' flashes maintain timing constraints
S6
Argument by testing with appropriate test cases
S7
Argument by testing with appropriate test cases
S8
Argument by testing with appropriate test cases
S9
Argument by testing with appropriate test cases
G21
Software test result for property 2 showed no violation for all modes
G22
Software test result for property 3 showed no violation for all modes
G23
Software test result for property 4 showed no violation for all modes
G24
Software test result for property 5 showed no violation for all modes
C13
Property 2 satisfied by the statement: while(timev.read_ms()>=avimin && timev.read_ms()<=avimax) (Ensuring a Vsense/Vpace between AVImin and AVImax)
Sn7
Test result evidenced during runtime trace
C14
Property 3 & 4 satisfied by the statement: while(timea. read_ms()<= lri [currentMode] - avimax - pvrp) and the flags if(asignal==1) adone =1; if(adone==0) apace =1;
Sn8
Test result evidenced during runtime trace
C15
Property 5 satisfied by the statement: while(timev.read_ms() <avimin); (Waiting for AVImin to commence)
Sn9
Test result evidenced during runtime trace

The assurance cases for timing tolerance are based on a goal that the synthesized code satisfies all the timing properties with tolerance of less than 1 millisecond. They are divided into subgoals in such a way that each of the property are satisfied . The properties are

★ Deadlock freeness

★ Ensuring Vsense/Vpace between AVImin and AVImax

★ Atrial sense receive confirmed after PVARP between URI and LRI

★ Atrial pace receive confirmed after LRI - PVARP - AVImax

★ Confirm Atrial sense or pace and putting heart in wait state for Vsense/Vpace till AVImin

They are supported with arguments that appropriate test cases were tested. The strategy used to prove these properties are the test results which illustrate the assured working of the test object.

Each of these properties are taken care by implementing guard conditions that are to be satisfied in such a way that they confine events that occur and events that are to be triggered, to their particular time interval respectively.

These test cases for specific properties are designed with a timing tolerance of ± 1 *millisecond* . They are tested for exact boundary conditions , a deviation of +1 millisecond and a deviation of 1 millisecond.

The observed result that the pacemaker paces the ventricle and the atrium as required provide justification.

The **correctness of implementation** is verified by:

1. Observing the heart rate on the LCD. Whenever the heart is in manual mode, the heart rate shifts towards the lower bound i.e. LRI.
2. Test Modes are implemented where the Pacemaker is tested rigorously.
3. Test Modes test the pacemaker for all the possible timing constraints.
4. Pacemaker always paces the heart when needed as it is implemented as a FSM(Finite State Machine). This implies the state changes when it needs to.
5. The implementation is done only after model checking. Since all the queries are satisfied and the model is assured to be deadlock free in UPPAAL, with the same timing constraints and the observed outputs, it can be confidently stated that the implementation is correct.

Member Contribution:

Every member of the project has worked equally towards modelling, development and implementation of the project. Everyone has shown full dedication towards working on this project.