Specification Report

Pain killer inject system

Team3

student xxxx

Content

S3.5 Receiving patient call

S3.6 Resolving patient call

S3.7 Show function of inject amount and time during this hour

S3.8 Show function of inject amount and time during this day

S3.9 Update power lamps, baseline and left_amount.

S3.10 Update current time

Starter implementation

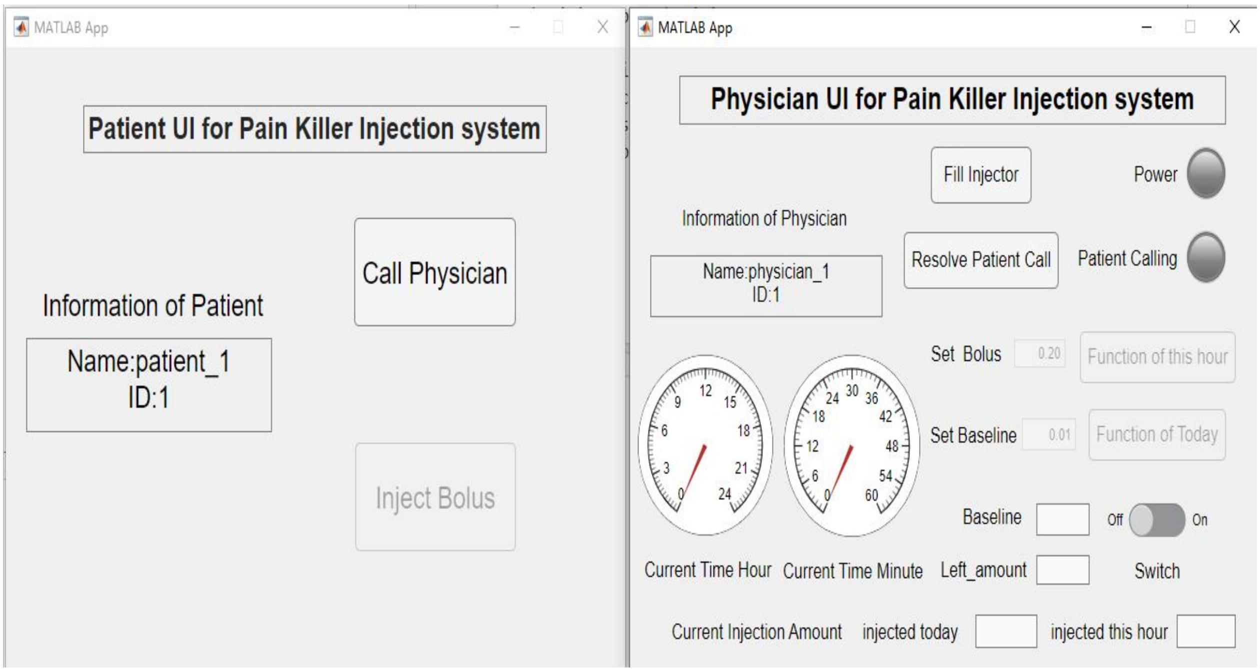
System Structure

Introduction of the system

Remark: To simulate the injection in reality, 1 sec in the program is equal to 1 min in reality. Since if we use the time in reality, the program may run a few hours, which is not efficient and time-consuming. Notice that all the time in the report is the time in the program instead of the reality.

The system is consist of 4 main component, pain killer controller, patient UI, physician UI and starter. Where pain killer controller is implemented in controller.m, patient UI is implemented in patient_UI.mlapp, physician UI is implemented in physician_UI.mlapp and starter is implemented in main.m.

You can launch the program by running main.m, which is the starter. Then you will see the patient UI and physician UI. So you can now operate on the 2 UI to use them. The following is what is appearing after starting the program.



Information of Patient

Name:patient_1
ID:1

Information of Physician

Name:physician_1
ID:1

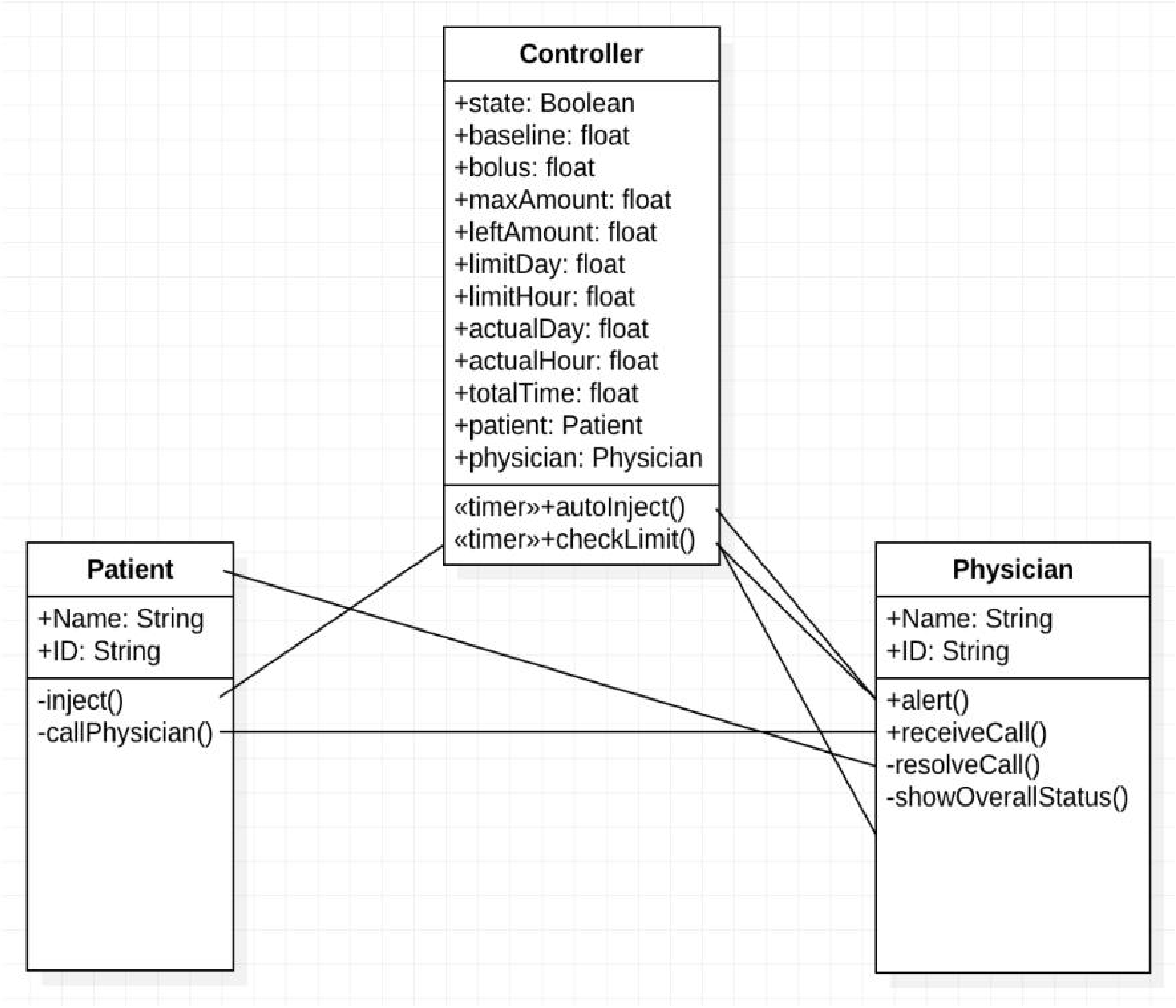
Set Bolus 0.20

Set Baseline 0.01

The specific properties, methods and how each part interact with others will be introduced in the following part.

Remark: Starter is not numbered S4, since it is just a interface and a way of launching the program and it serves no concrete functional propose. Consequently, starter will not match anything in requirement and validation. So it should not be numbered due to traceability.

The system structure is shown below



S1: pain killer controller implementation

Properties introduction

The following are properties in controller.m. In the left side is the name and initial value of the properties(some does not have a initial value since they will be evaluated later) On the right side is the description of the properties(after %).

Code:

```
state=0;                      % state of the injector, 0 off, 1 on
Baseline=0.01;                % medicine injected automatically per minute,
                              0.01-0.1
Bolus=0.2;                     % medicine injected manually per shoot,0.2-0.5
Amount=10;                     % total capacity of injector
Left_amount=10;                % left amount of injector
limit_per_day=3;               % limit per day
exceed_limit_day=0;            % state of whether day limit exceed,0 not exceed,1
                               exceed
limit_per_hour=1;              % limit per hour
exceed_limit_hour=0;           % state of whether hour limit exceed,0 not exceed,1
                               exceed
hour=0;                        % hour passed since    the first time injector is on
day_injected=0;                 % total injection today
minute=0;                      % minute passed since the first time injector is on
hour_injected=0;                % total injection this hour
```

```
t                              % control injection
t1                             % count hour and minute
patient_UI                     % patient_UI.mlapp
physician_UI                   % physician_UI.mlapp
minute_count=0                 % decide the function to calculate minute and hour
minute_elapsed=0               % the minute passes since the first time injector is on
                               start, >=60    >=14400
hour_vec=ones(1,600)-1      % store every minute in an hour
func_hour_vec=ones(1,14400)-1      % store inject amount each hour(cumulated)
day_vec=ones(1,14400)-1            % store every minute in a day
func_day_vec=ones(1,14400)-1       % store inject amount each day(cumulated)
hour_vec_ptr=1                % store index of element in hour_vec (1<=ptr<=600)
func_hour_vec_ptr=1;          % store index of element in func_hour_vec
                              (1<=ptr<=14400)
day_vec_ptr=1                 % store index of element in day_vec
                              (1<=ptr<=14400)
func_day_vec_ptr=1;           % store index of element in func_day_vec
                              (1<=ptr<=14400)
auto_inject_on=0              % decide whetjer auto inject is on,1 is on,0 is off
```

S1.1 Set timer

There are 2 timer, t and t1. t is use for controlling the automatic injection of baseline, which will be specified in S1.2. Function in S1.2 is the TimerFuc of t,so it will be called once per period, in this specific case 0.1 sec in reality(0.1 min in program). t1 is used for controlling limit and calculating time, which will be specified in S1.3 and S1.4. Function in S1.3 and S1.4 is the TimerFuc of t1 ,so it will be called once per period, in this specific case 0.1 sec in reality(0.1 min in program).

t will be on when the injector is on and it is not empty, and be off otherwise. t1 will be on since the first time the injector is on, and be off before it is on. It will never be turned off if it is once on.

The parameter of the 2 timers is defined in the construction function of the class, in this case controller().

Code

```
function obj=controller()
           obj.t=timer;
           obj.t.TimerFcn=@obj.auto_inject;
           obj.t.Period=0.1;
           obj.t.ExecutionMode='fixedRate';
           obj.t1=timer;
           obj.t1.TimerFcn=@obj.limit_control;
           obj.t1.Period=0.1;
           obj.t1.ExecutionMode='fixedRate';
       end
```

S1.2 control inject baseline

As stated in S1.1,S1.2 is the timer function of t in S1.1. And it is used for controlling the automatic injection. The Timerfcn is auto_inject.

It will first check whether the injector is on

If injector is off, do nothing.

If injector is on, check if the there is any drug in injector

If it is empty, stop timer t and set left amount to 0.

If it is not empty, inject baseline.

Finally call ChangeState to update information in Physician UI, which will be specifiec in S3.9.

Code:

```
% medicine injected automatically per 0.1 minute
        function auto_inject(Obj,~,~)
            switch Obj.state
                case 0  % injector off
                case 1  % injector on
                    if (Obj.Left_amount<=0)  % when empty,stop timer but injector is still on
                        stop(Obj.t);
                        Obj.Left_amount=0;
                    else    % when non-empty,inject and Left_amount decrease
                        if ((Obj.exceed_limit_day==0)&&(Obj.exceed_limit_hour==0))    % not exceed day limit or hour limit
                            Obj.Left_amount=Obj.Left_amount-Obj.Baseline*0.1;
                        end
                    end
            end
            Obj.physician_UI.changeState(Obj.state,Obj.Baseline,Obj.Left_amount);   %change message and lamp in physician_UI
        end
```

S1.3 control not to exceed day limit and hour limit

As stated in S1.1. S1.3 and S1.4 is the timer function of t1 in S1.1. And t1 is used for controlling limit and calculating time. The Timerfcn is limit_control.

First it check whether the limit of hour exceed, limit of day exceed and whether the auto_inject should be on.

Then it update the index of func_hour_vec and func_day_vec, the 2 function will be used for plotting, which will be specified in S3.7 and S3.8.

After that it will update hour_vec, day_vec(store the data of inject amount of every minute, each in an element of the array). And also update hour_injected and day_injected. How to calculate these data is decided by whether injector is on and how many time has passed since the t1 has on.

Then it update the index of the array used to store how much has been injected this hour.

Code:

```
% control not to exceed day limit and hour limit
       function limit_control(Obj,~,~)
           Obj.minute_elapsed=Obj.minute_elapsed+0.1;
           Obj.exceed_limit_hour=0;
           Obj.exceed_limit_day=0;
           Obj.auto_inject_on=0;
           %judge whether exceed limit
           if (Obj.hour_injected>=Obj.limit_per_hour-Obj.Baseline*0.1) % exceed hour limit
               Obj.exceed_limit_hour=1;
           end
           if (Obj.day_injected>=Obj.limit_per_day-Obj.Baseline*0.1) % exceed hour limit
               Obj.exceed_limit_day=1;
           end
           %judge whether auto inject is on
           if
((Obj.state==1)&&(Obj.Left_amount>0)&&(Obj.exceed_limit_day==0)&&(Obj.exceed_limit_hour==0))
               Obj.auto_inject_on=1;
           end
           % store func_hour_vec and func_day_vec
           if (Obj.minute_elapsed>0.1)
               Obj.func_hour_vec(Obj.func_hour_vec_ptr+1)=Obj.hour_injected;
               Obj.func_day_vec(Obj.func_day_vec_ptr+1)=Obj.day_injected;
           end
```

```
% index out of range,reset index
Obj.func_hour_vec_ptr=Obj.func_hour_vec_ptr+1;
Obj.func_day_vec_ptr=Obj.func_day_vec_ptr+1;
if (Obj.func_hour_vec_ptr>14400)
    Obj.func_hour_vec_ptr=1;
end
if (Obj.func_day_vec_ptr>14400)
    Obj.func_day_vec_ptr=1;
end
% store amount this minute and calculate hour_injected and
% day_injected
if(Obj.auto_inject_on==1)
    if(Obj.minute_elapsed>1440)
        % when sub,only need to sub Obj.hour_vec(Obj.hour_vec_ptr)
        % and Obj.day_vec(Obj.day_vec_ptr) since they contain both
        % auto_inject and Bolus
        Obj.hour_injected=Obj.hour_injected-Obj.hour_vec(Obj.hour_vec_ptr);
        Obj.day_injected=Obj.day_injected-Obj.day_vec(Obj.day_vec_ptr);
        % initially,contains only baselin,so later Bolus will be
        % added to it
        Obj.hour_vec(Obj.hour_vec_ptr)=Obj.Baseline*0.1;
        Obj.day_vec(Obj.day_vec_ptr)=Obj.Baseline*0.1;
        % when add,need to add Obj.hour_vec(Obj.hour_vec_ptr)
        % and Obj.day_vec(Obj.day_vec_ptr) since they contain
        % auto_inject .So when Bolus, Obj.hour_vec(Obj.hour_vec_ptr)
        % and Obj.day_vec(Obj.day_vec_ptr) and also hour_injected
        % and day_injected all need to be added
        Obj.hour_injected=Obj.hour_injected+Obj.hour_vec(Obj.hour_vec_ptr);
        Obj.day_injected=Obj.day_injected+Obj.day_vec(Obj.day_vec_ptr);
    elseif((Obj.minute_elapsed>60)&&(Obj.minute_elapsed<=1440))
        Obj.hour_injected=Obj.hour_injected-Obj.hour_vec(Obj.hour_vec_ptr);
        Obj.hour_vec(Obj.hour_vec_ptr)=Obj.Baseline*0.1;
        Obj.day_vec(Obj.day_vec_ptr)=Obj.Baseline*0.1;
        Obj.hour_injected=Obj.hour_injected+Obj.hour_vec(Obj.hour_vec_ptr);
        Obj.day_injected=Obj.day_injected+Obj.day_vec(Obj.day_vec_ptr);
    else
        Obj.hour_vec(Obj.hour_vec_ptr)=Obj.Baseline*0.1;
        Obj.day_vec(Obj.day_vec_ptr)=Obj.Baseline*0.1;
        Obj.hour_injected=Obj.hour_injected+Obj.hour_vec(Obj.hour_vec_ptr);
        Obj.day_injected=Obj.day_injected+Obj.day_vec(Obj.day_vec_ptr);
    end
else
    if(Obj.minute_elapsed>1440)
        Obj.hour_injected=Obj.hour_injected-Obj.hour_vec(Obj.hour_vec_ptr);
        Obj.day_injected=Obj.day_injected-Obj.day_vec(Obj.day_vec_ptr);
        Obj.hour_vec(Obj.hour_vec_ptr)=0;
        Obj.day_vec(Obj.day_vec_ptr)=0;
    elseif((Obj.minute_elapsed>60)&&(Obj.minute_elapsed<=1440))
        Obj.hour_injected=Obj.hour_injected-Obj.hour_vec(Obj.hour_vec_ptr);
```

```
                    Obj.hour_vec(Obj.hour_vec_ptr)=0;
                    Obj.day_vec(Obj.day_vec_ptr)=0;
                else
                    Obj.hour_vec(Obj.hour_vec_ptr)=0;
                    Obj.day_vec(Obj.day_vec_ptr)=0;
                end

            end
            %move to next index
            Obj.hour_vec_ptr=Obj.hour_vec_ptr+1;
            Obj.day_vec_ptr=Obj.day_vec_ptr+1;
            % if exceed the limit,go to begin of vec
            if (Obj.hour_vec_ptr>600)
                Obj.hour_vec_ptr=1;
            end
            if (Obj.day_vec_ptr>14400)
                Obj.day_vec_ptr=1;
            End

Obj.physician_UI.showTime(Obj.hour,Obj.minute,Obj.day_injected,Obj.hour_injected);   %change message  in physician_UI
        end
```

S1.4 calculate current time

Simple code for calculating hour many hour and minute has passed since t1 is on.

Code

```
% calculate time
            Obj.minute=Obj.minute+0.1;
            if( Obj.minute>=60)
                Obj.minute=0;
                Obj.hour=Obj.hour+1;
                if(Obj.hour>=24)
                    Obj.hour=0;
                end
            end
```

S2: patient UI implementation

Properties introduction

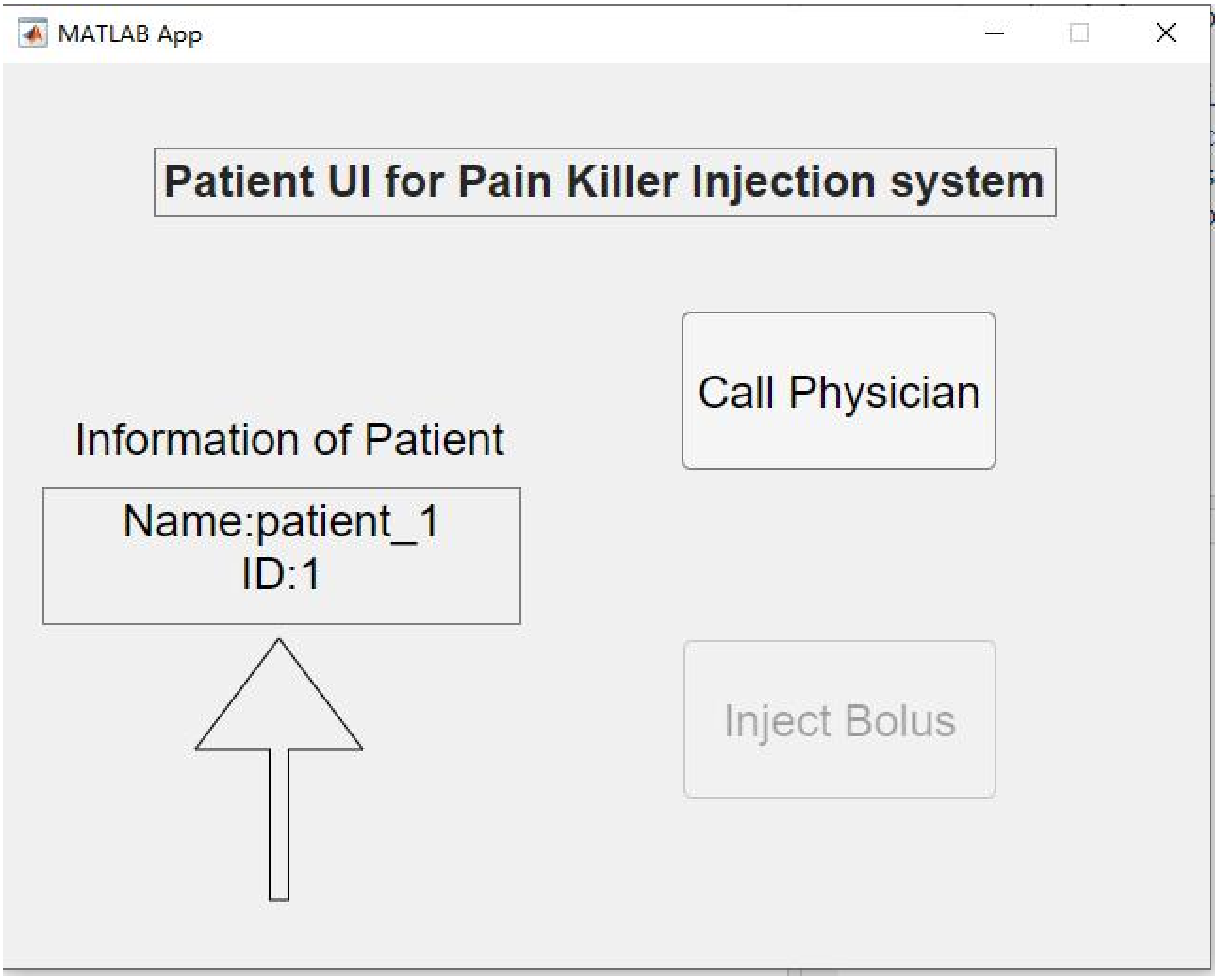
The following are properties in patient_UI.mlapp. In the left side is the name and initial value of the properties(some does not have a initial value since they will be evaluated later) On the right side is the description of the properties(after %).

Code:

```
properties (Access = public)
        controller                        % controller.m
        Name='patient_1';                 % patient Name
        ID='1';                             % patient ID
        error3;                             % Error code,Insufficient left amount
        error4;                             % Error code,hour injection limit or day
                                        injection limit will or has already exceeded
end
```

S2.1 Show patient name and ID

Print the patient name and ID in the specified area InformationofPatientTextArea.



Patient UI for Pain Killer Injection system

Information of Patient

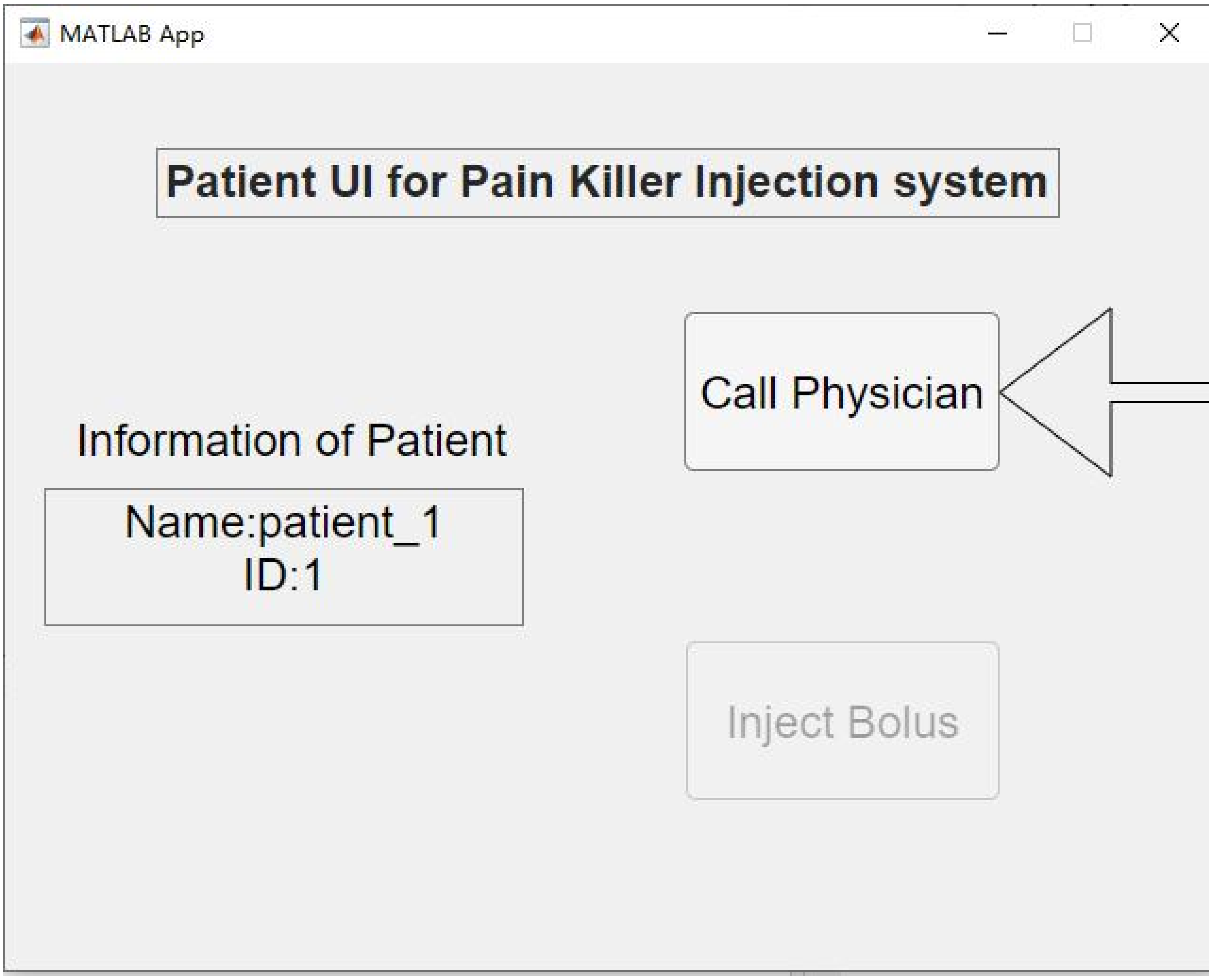
Name:patient_1
ID:1

Code:

```
  function startupFcn(app)% print patient information
     name_id=sprintf('Name:%s\nID:%s',app.Name,app.ID);
     app.InformationofPatientTextArea.Value=name_id;
  end
```

S2.2 Call physician

If the patient has any problem or emergencies, the patient can call the physician for help. The physician will see PatientCallingLamp turning red, the detail will be specified in S3.5 and S3.6.



Code:

```
function CallPhysicianButtonPushed(app, event)
    app.controller.physician_UI.patientCall;
end


function patientCall(app)
    app.PatientCallingLamp.Color=[1,0,0];
end
```

S2.3 Inject bolus

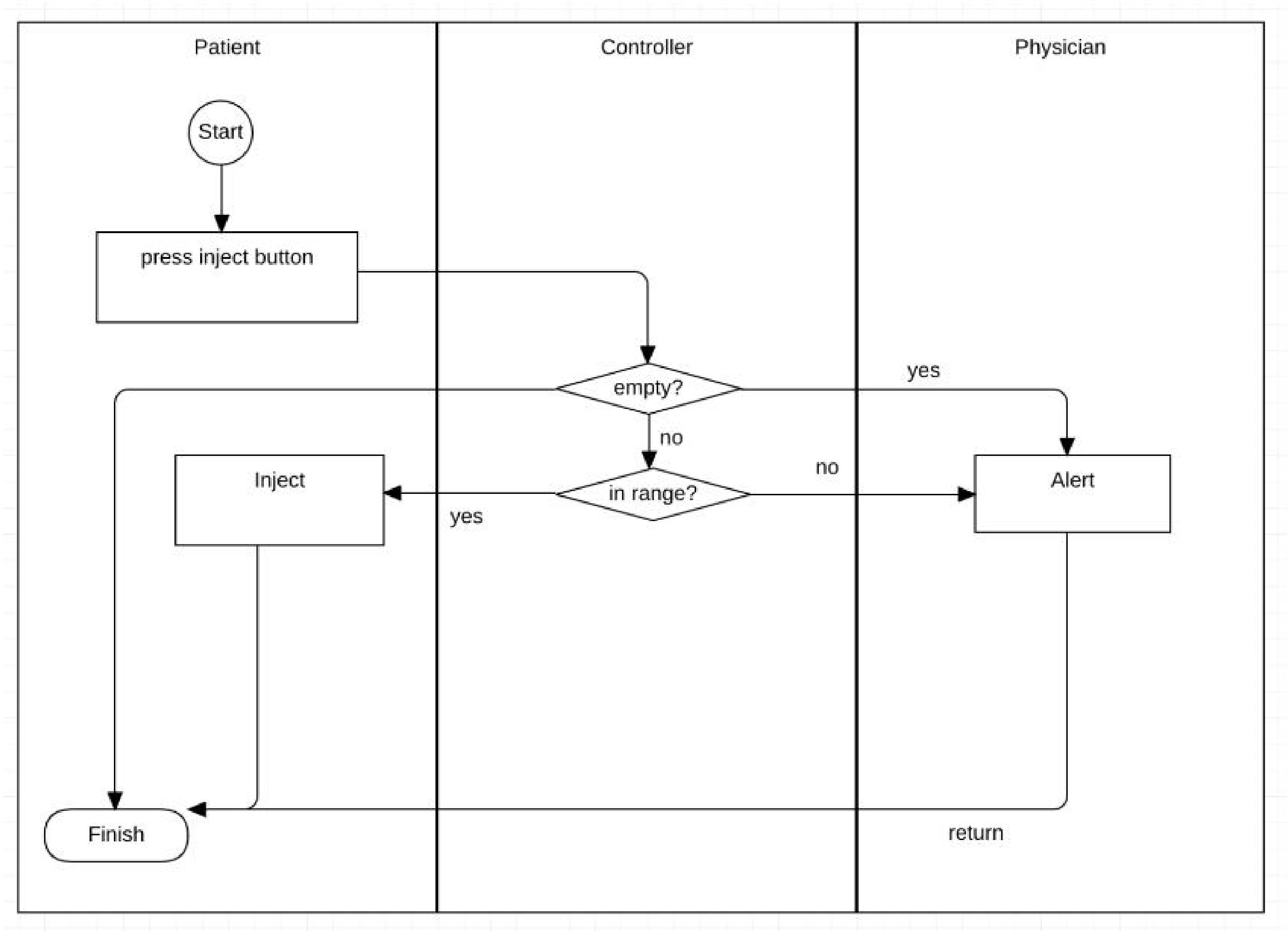
If the patient feels extremely painful, the patient can press inject Bolus button to inject some drug. The button is enabled only when injector is on.

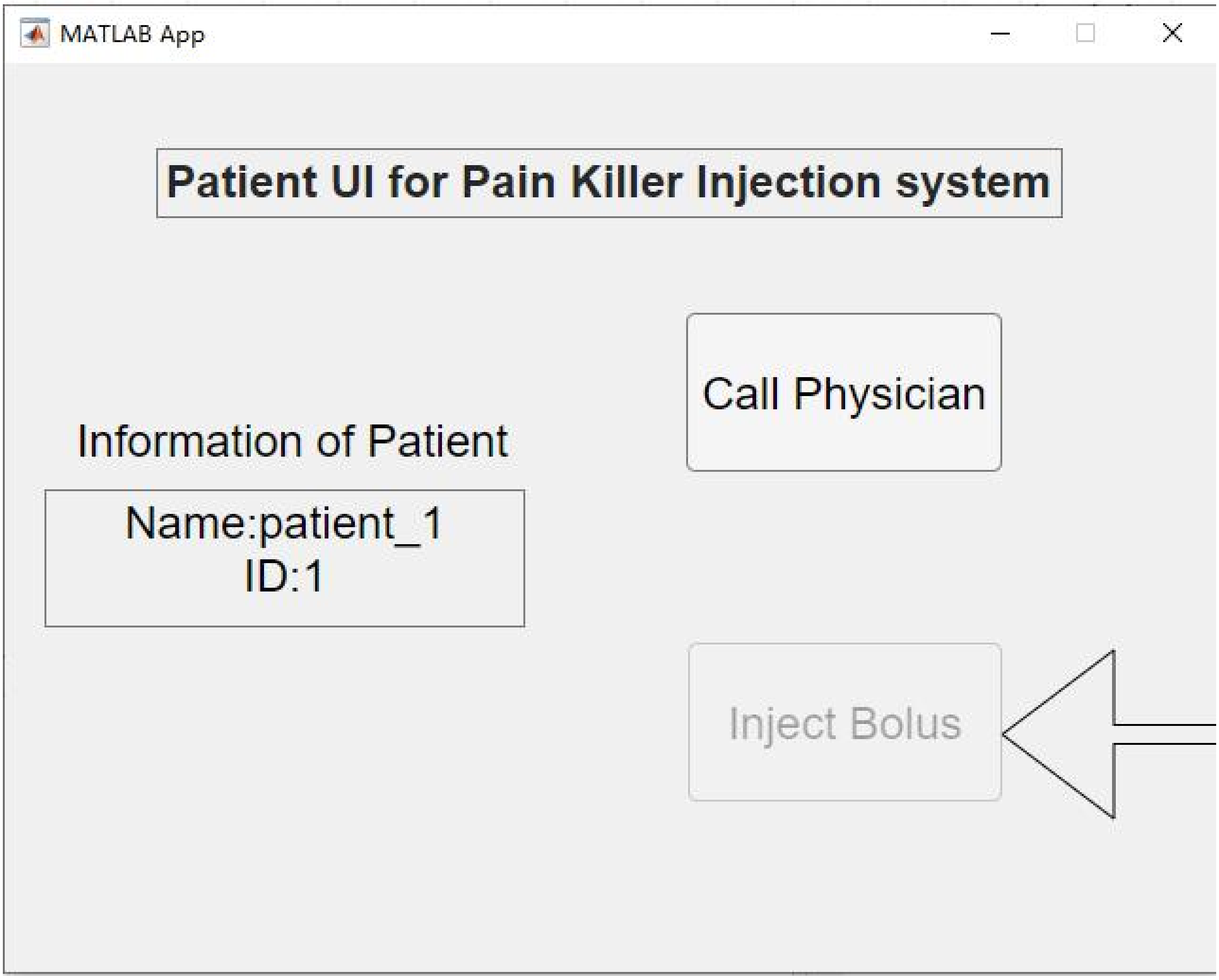
The function will first check whether the injection is valid,if the injector is on and there is adequate drug and the limit of day and hour will not exceed before and after the injection.

If all the requirement is met, it will update day_injected, hour_injected, left_amount and the vectors that stores data.

Else, it will create an message box with corresponding error message.

Patients giving themselves a single bolus:





Name:patient_1
ID:1

Code

```
function InjectBolusButtonPushed(app, event)
           % inject a Bouls (disabled when injector is off)
           if (app.controller.state==1)  % injector on
               if ((app.controller.Bolus<=app.controller.Left_amount) &&
(app.controller.exceed_limit_day==0) && (app.controller.exceed_limit_hour==0) &&
(app.controller.hour_injected+app.controller.Bolus<app.controller.limit_per_hour-app.controller.
Baseline*0.1) &&
(app.controller.day_injected+app.controller.Bolus<app.controller.limit_per_day-app.controller.B
aseline*0.1))
                  app.controller.Left_amount=app.controller.Left_amount-app.controller.Bolus;

app.controller.hour_injected=app.controller.hour_injected+app.controller.Bolus;

app.controller.day_injected=app.controller.day_injected+app.controller.Bolus;

app.controller.hour_vec(app.controller.hour_vec_ptr-1)=app.controller.hour_vec(app.controller.h
our_vec_ptr-1)+app.controller.Bolus;

app.controller.day_vec(app.controller.day_vec_ptr-1)=app.controller.day_vec(app.controller.day_
vec_ptr-1)+app.controller.Bolus;
                  left=sprintf('%.2f',app.controller.Left_amount);
```

```
                    app.controller.physician_UI.Left_amountTextArea.Value=left;
                elseif (app.controller.Bolus>app.controller.Left_amount)  % Error
                    app.error3 = msgbox('Error code 3,Insufficient left amount', 'Error','error');
                else    % Error
                    app.error4 = msgbox('Error code 4,hour injection limit or day injection limit
will or has already exceeded.', 'Error','error');
                end
            end
        end
```

S3: physician UI implementation

Properties introduction

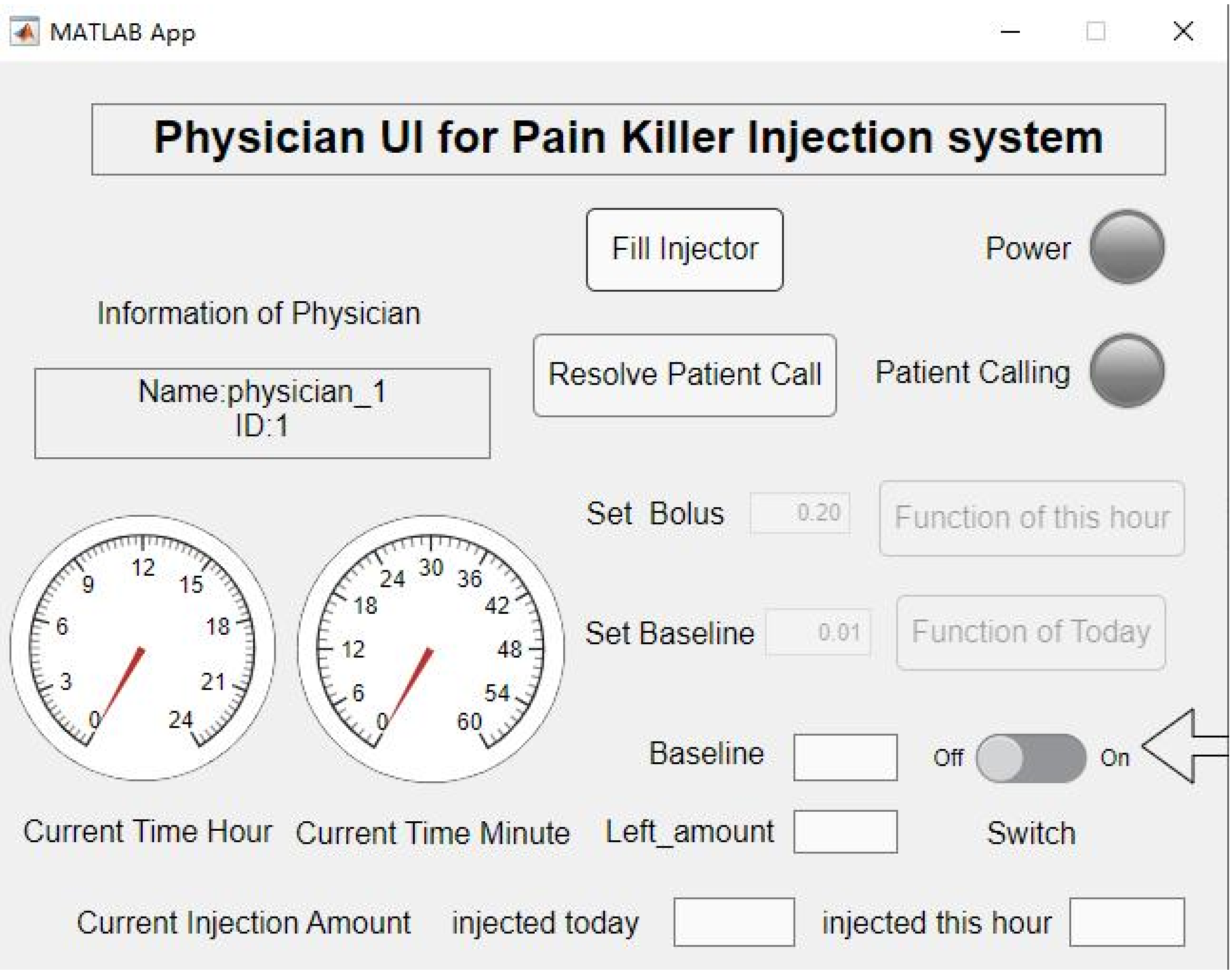
The following are properties in physician_UI.mlapp. In the left side is the name and initial value of the properties(some does not have a initial value since they will be evaluated later) On the right side is the description of the properties(after %).

```
    properties (Access = public)
        controller            % controller.m
        Name='physician_1';     % physician Name
        ID='1';               % physician ID
        first_on=0;           % state whether injector has on at least once,1 yes 0 no
        old_Baseline=0.01;
        old_Bolus=0.2;
        error1;
        error2;
        hour_figure;
        day_figure;
    end
```

S3.1 handle switch on and off

The switch for injector has 2 state, on and off. When on, all the function is enabled and the power light will turn green. When off, those function related to injection will be disabled , such as inject bolus, inject baseline, set bolus and set baseline and the light will turn grey. While other function can still work.

It will also start t if there is enough drug and start t1 if it is the first time injector is on.



Name:physician_1
ID:1

Set Bolus 0.20

Set Baseline 0.01

Code:

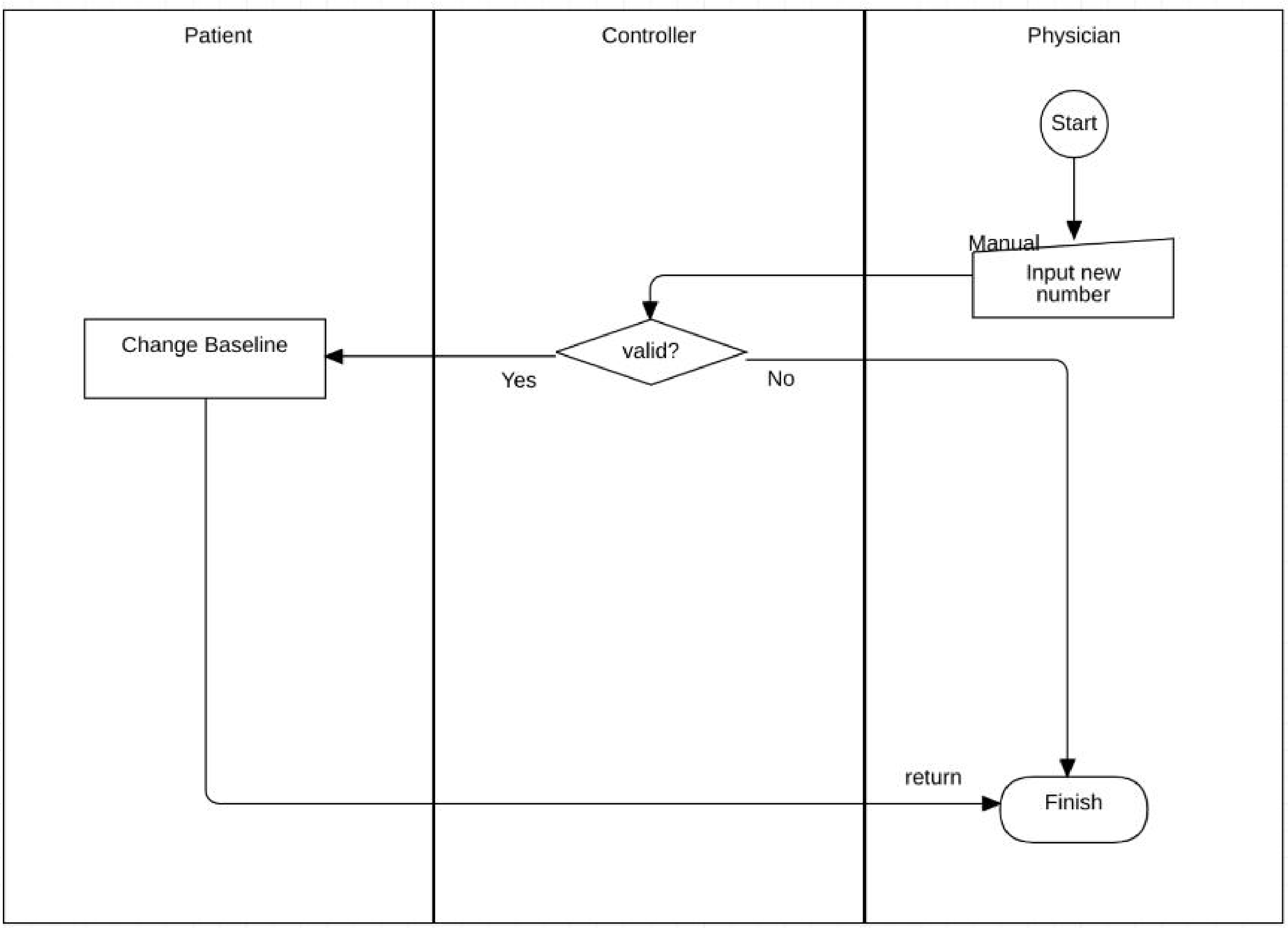
```
function SwitchValueChanged(app, event)
           % switch of injector
           value = app.Switch.Value;
           if(strcmp('Off',value)==1)
               stop(app.controller.t);
               app.controller.state=0;
               app.PowerLamp.Color=[0.5,0.5,0.5];
               app.FunctionofTodayButton.Enable="off";
               app.FunctionofthishourButton.Enable="off";
               app.SetBaselineEditField.Editable="off";
               app.SetBaselineEditField.Enable="off";
               app.SetBolusEditField.Editable="off";
               app.SetBolusEditField.Enable="off";
               app.controller.patient_UI.InjectBolusButton.Enable="off";
           elseif(strcmp('On',value)==1)
               app.FunctionofTodayButton.Enable="on";
               app.FunctionofthishourButton.Enable="on";
               app.SetBaselineEditField.Enable="on";
```

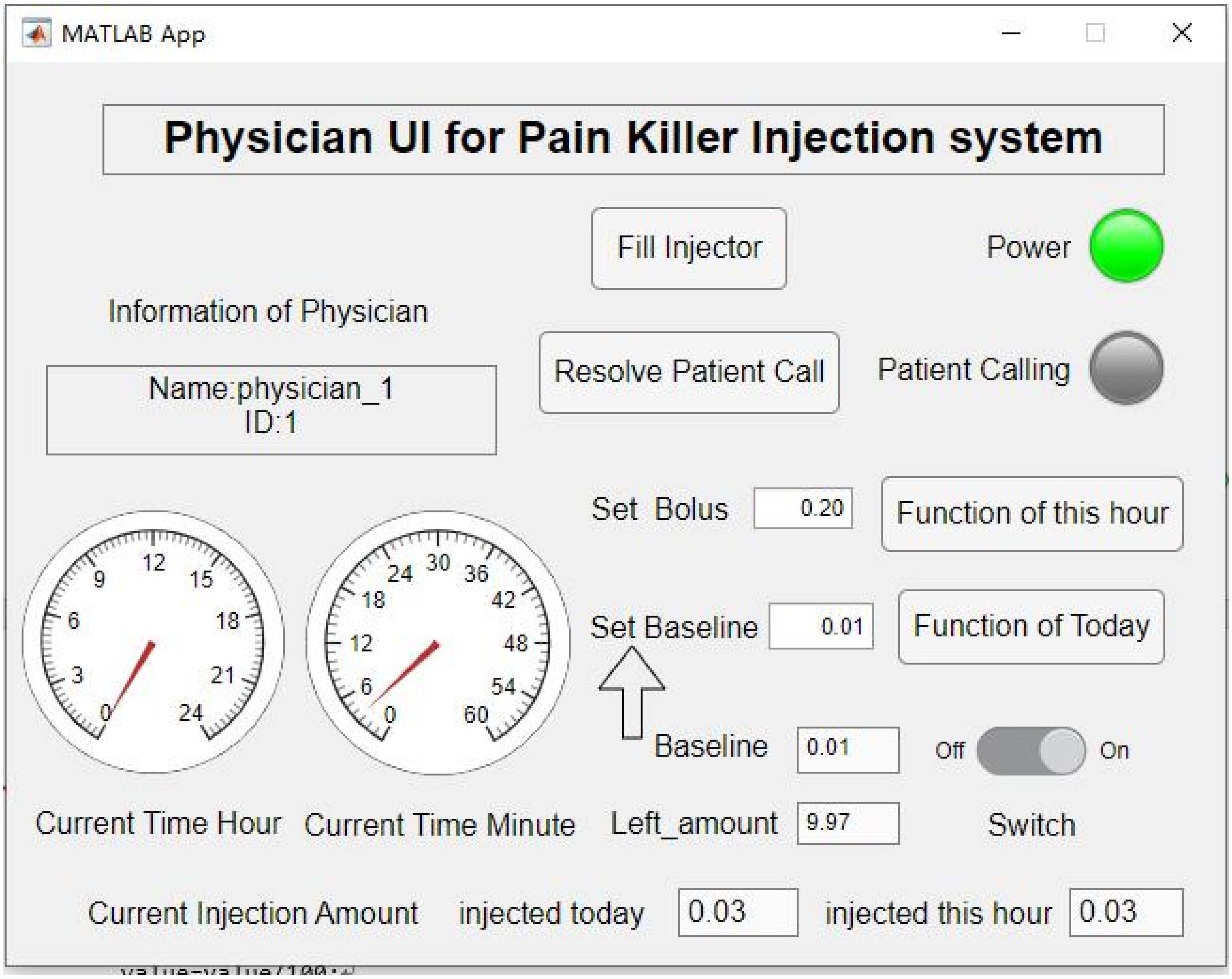
```
                app.SetBaselineEditField.Editable="on";
                app.SetBolusEditField.Enable="on";
                app.SetBolusEditField.Editable="on";
                app.controller.patient_UI.InjectBolusButton.Enable="on";
                if (app.controller.Left_amount>0) % not empty
                    start(app.controller.t);
                end
                if app.first_on==0 % first time injector is on,start timer t1
                    start(app.controller.t1);
                end
                app.controller.state=1;
                app.first_on=1;
                app.PowerLamp.Color=[0,1,0];
            end
        end
```

S3.2 Set Baseline

Physician can set the baseline to adjust the speed of auto injection by typing umber into the SetBaselineEditField.

Physicians setting a new value (as baseline):





Information of Physician

Name:physician_1
ID:1

Set Bolus 0.20

Set Baseline 0.01

Baseline 0.01

Left_amount 9.97

Current Injection Amount injected today 0.03 injected this hour 0.03

First, it judge if the amount is within the limit, in this case 0.01ml to 0.1ml.

If out of the range, create message box with error message.

If in the range, change the baseline

Code:

```
function SetBaselineEditFieldValueChanged(app, event)
    % set value of Baseline
    value = app.SetBaselineEditField.Value;
    if (value<0.01 || value>0.1)  % Error( non-numberical input is handled by matlab already)
        app.error2 = msgbox('Error code 2,Baseline should between 0.01 and 0.1',
'Error','error');
        app.SetBaselineEditField.Value=app.old_Baseline;
    else    % valid input
        value=value*100;
        value=round(value);
        value=value/100;
```

```
            app.controller.Baseline=value;
            app.old_Baseline=value;
            app.SetBaselineEditField.Value=app.controller.Baseline;
        end
    end
```

S3.3 Set Bolus

Set baseline is similar to set baseline logically.

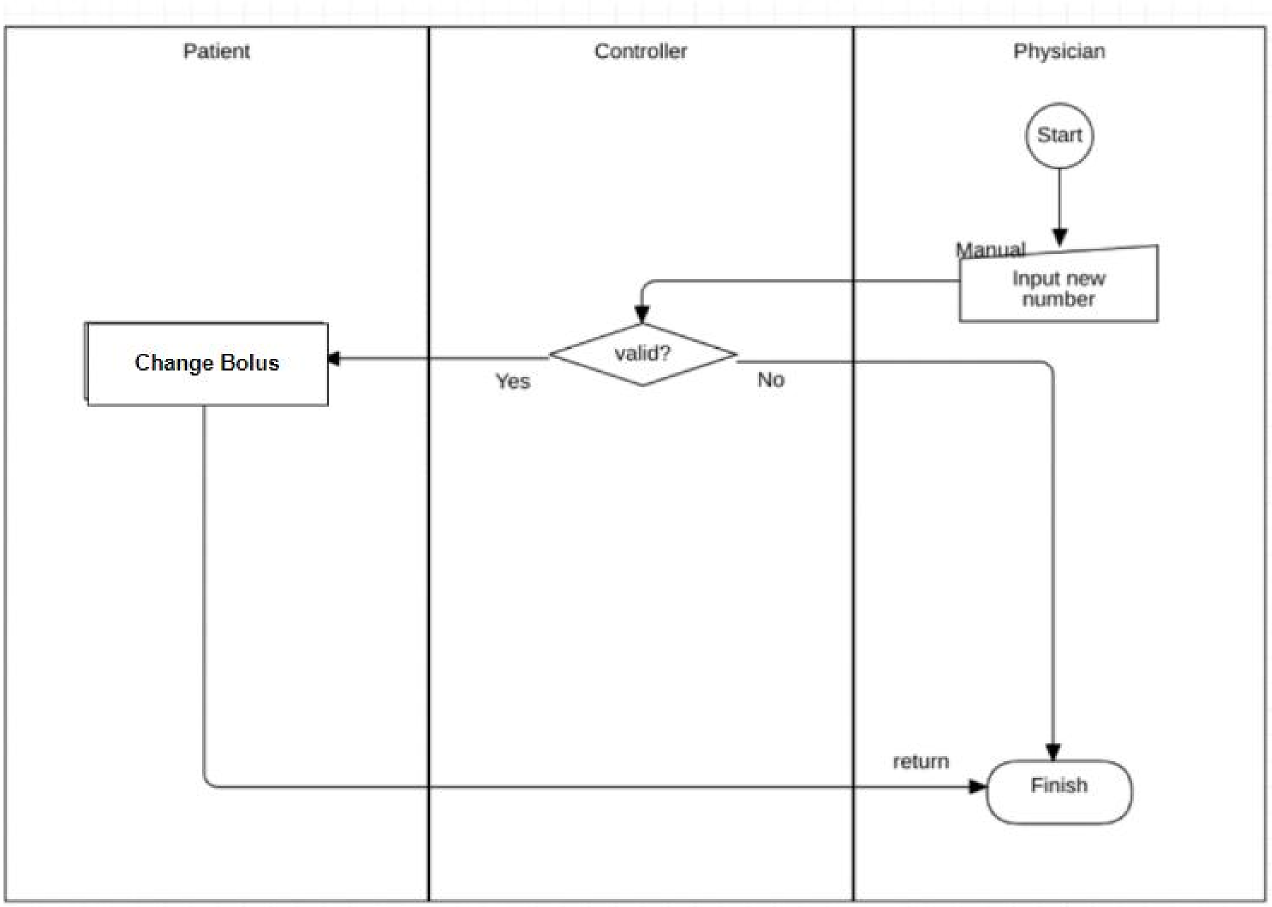
Physician can set the bolus by typing umber into the SetBolusEditField.

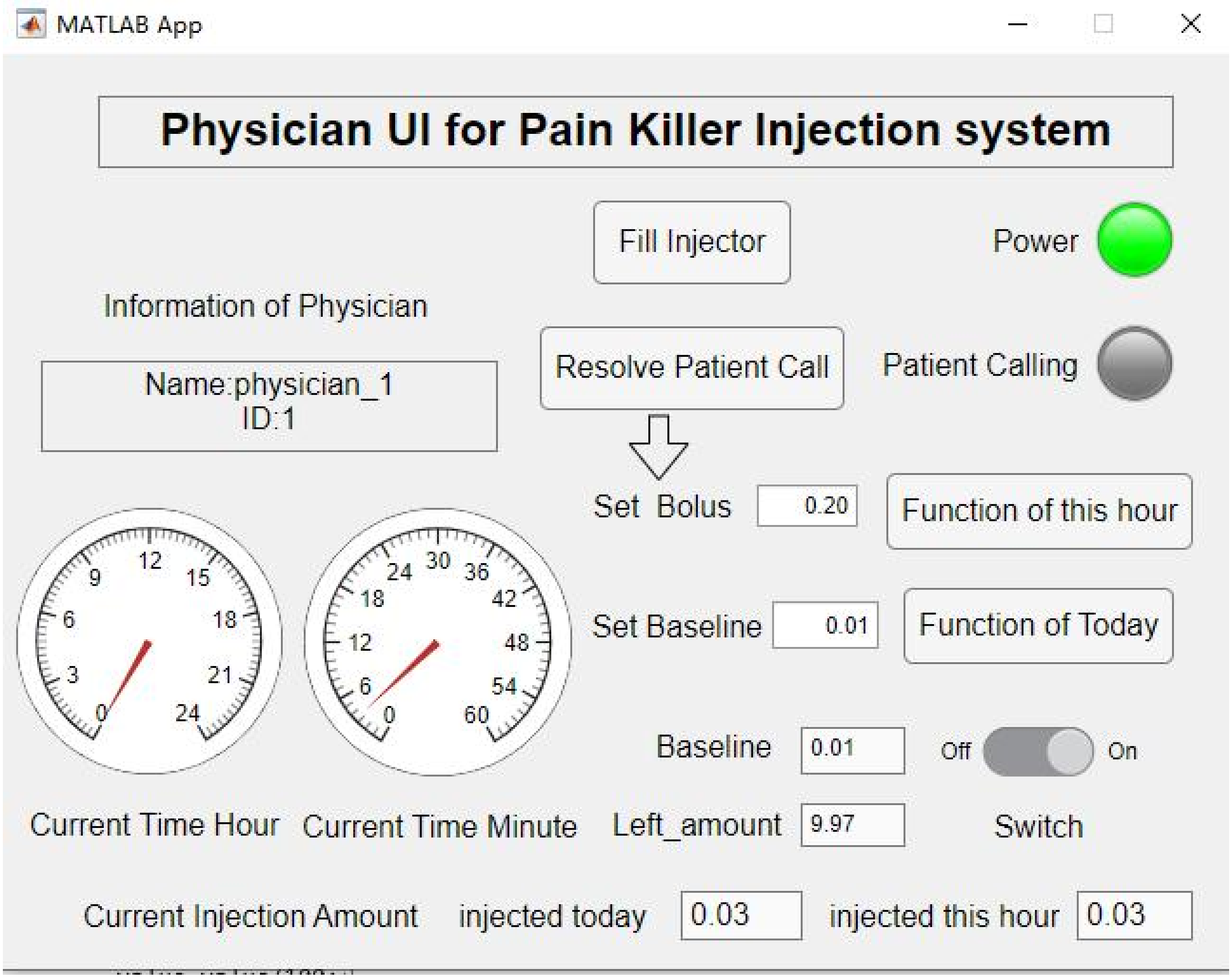
First, it judge if the amount is within the limit, in this case 0.2ml to 0.5ml.

If out of the range, create message box with error message.

If in the range, change the bolus

Physicians setting a new value (as bouls):





Physician UI for Pain Killer Injection system

Information of Physician

Name:physician_1
ID:1

Set Bolus 0.20

Set Baseline 0.01

Baseline 0.01

Current Time Hour Current Time Minute Left_amount 9.97

Current Injection Amount injected today 0.03 injected this hour 0.03

Code:

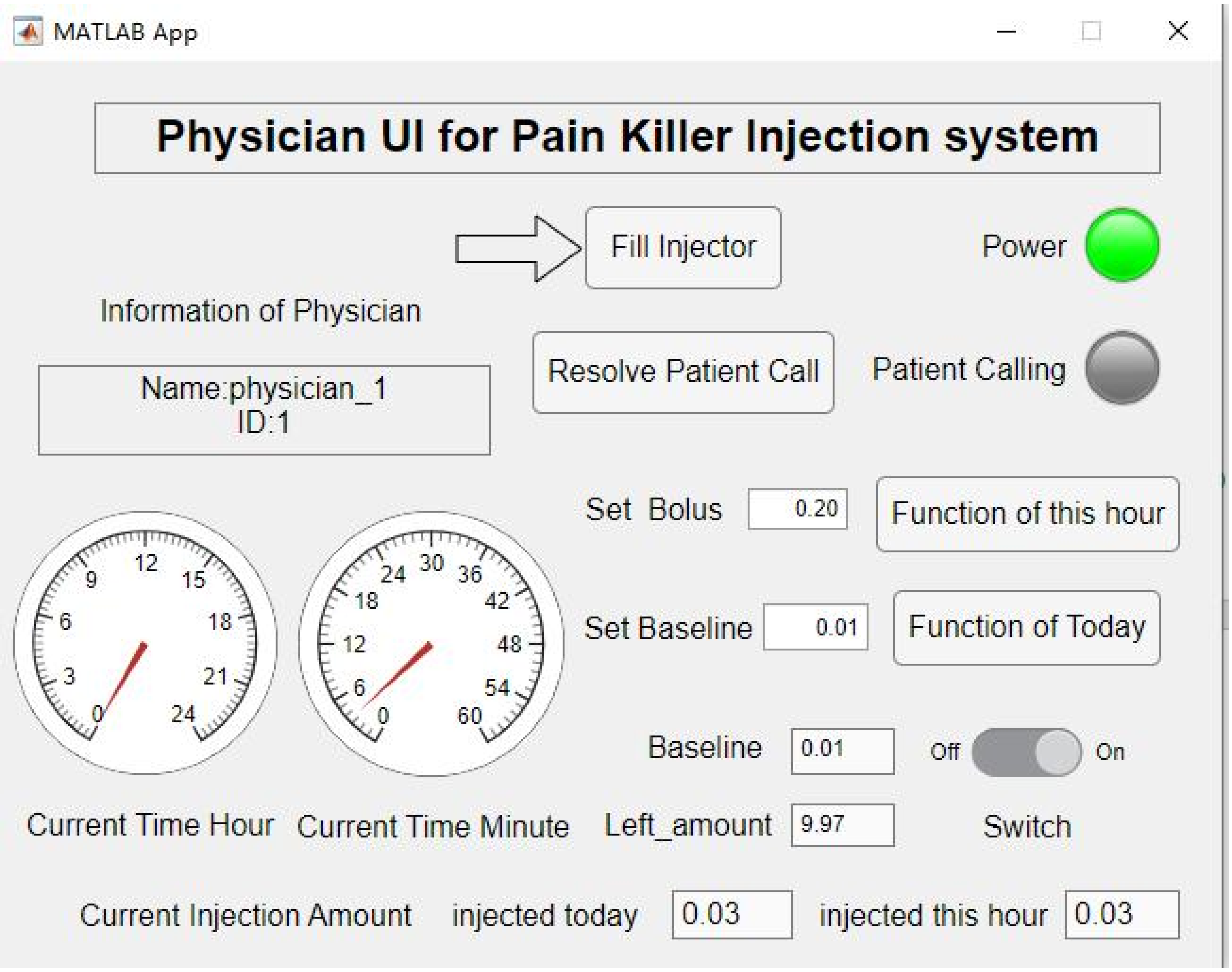
```
        function SetBolusEditFieldValueChanged(app, event)
            % set value of Bolus
            value = app.SetBolusEditField.Value;

            if (value<0.2 || value>0.5)    % Error( non-numberical input is handled by matlab already)
                 app.error1 = msgbox('Error code 1,Bolus should between 0.2 and 0.5',
'Error','error');
                 app.SetBolusEditField.Value=app.old_Bolus;
            else    % valid input
                value=value*100;
                value=round(value);
                value=value/100;
                app.controller.Bolus=value;
                app.old_Bolus=value;
                app.SetBolusEditField.Value=app.controller.Bolus;
            end
        end
```

S3.4 Fill Injector

Physician can use fill injector to add drugs into injector until it reaches maximum amount.

After filling, it will open t if necessary and update the Left_amountTextArea showing left_amount.



Name:physician_1
ID:1

Set Bolus 0.20

Set Baseline 0.01

Baseline 0.01

Left_amount 9.97

injected today 0.03

injected this hour 0.03

Code:

```
function FillInjectorButtonPushed(app, event)
    % fill the injector
    if app.controller.Left_amount<=0    % empty
        app.controller.Left_amount=app.controller.Amount;
        start(app.controller.t);
    else    % non-empty
```

```
        app.controller.Left_amount=app.controller.Amount;
    end
    left=sprintf('%.2f',app.controller.Left_amount);
    app.Left_amountTextArea.Value=left;
end
```

S3.5 Receiving patient call

After receiving the call the call, the PatientCallingLamp will turn red.

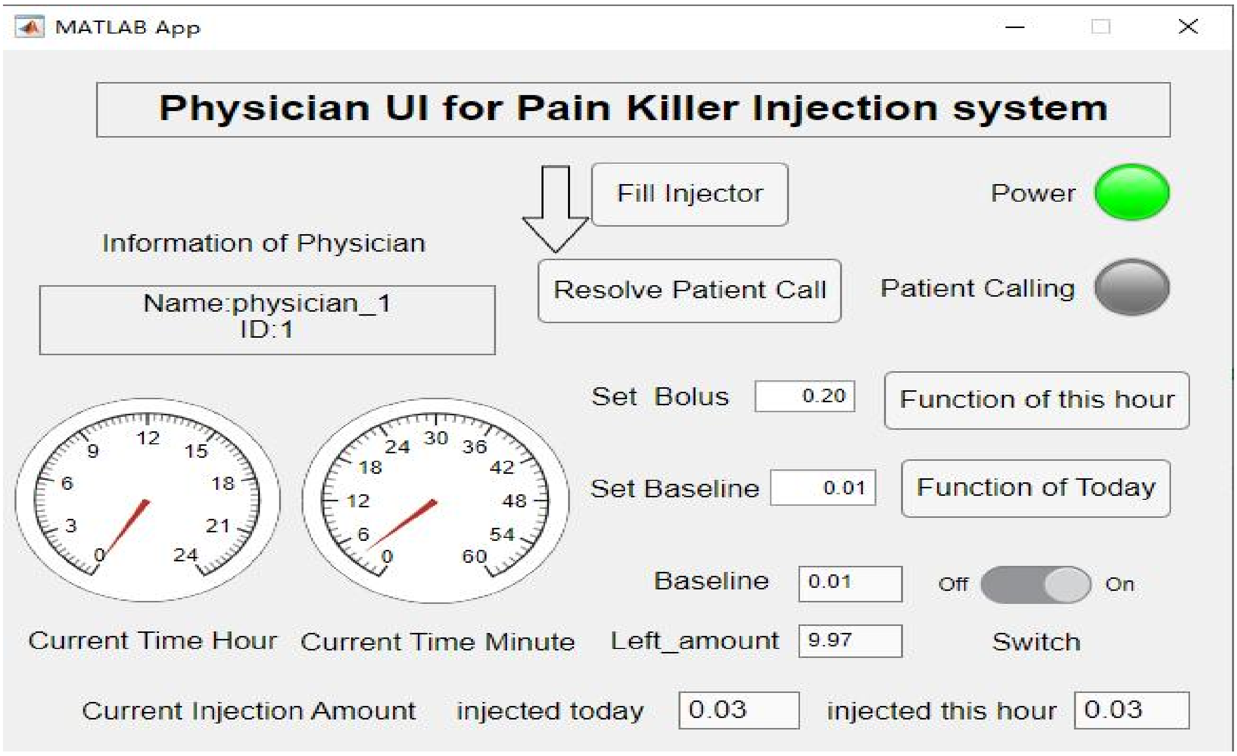
Which is also explained in S1.4.

Code:

```
% indicate a patinet call, change color of patient call lamp to red
function patientCall(app)
    app.PatientCallingLamp.Color=[1,0,0];
end
```

S3.6 Resolving patient call

Physician can resolve the call by pushing the ResolvePatientCallButton to tell the patient that he has receive the message he will be coming soon.



Physician UI for Pain Killer Injection system

Information of Physician

Name:physician_1
ID:1

Set Bolus 0.20

Set Baseline 0.01

Baseline 0.01

Left_amount 9.97

Current Injection Amount

injected today 0.03

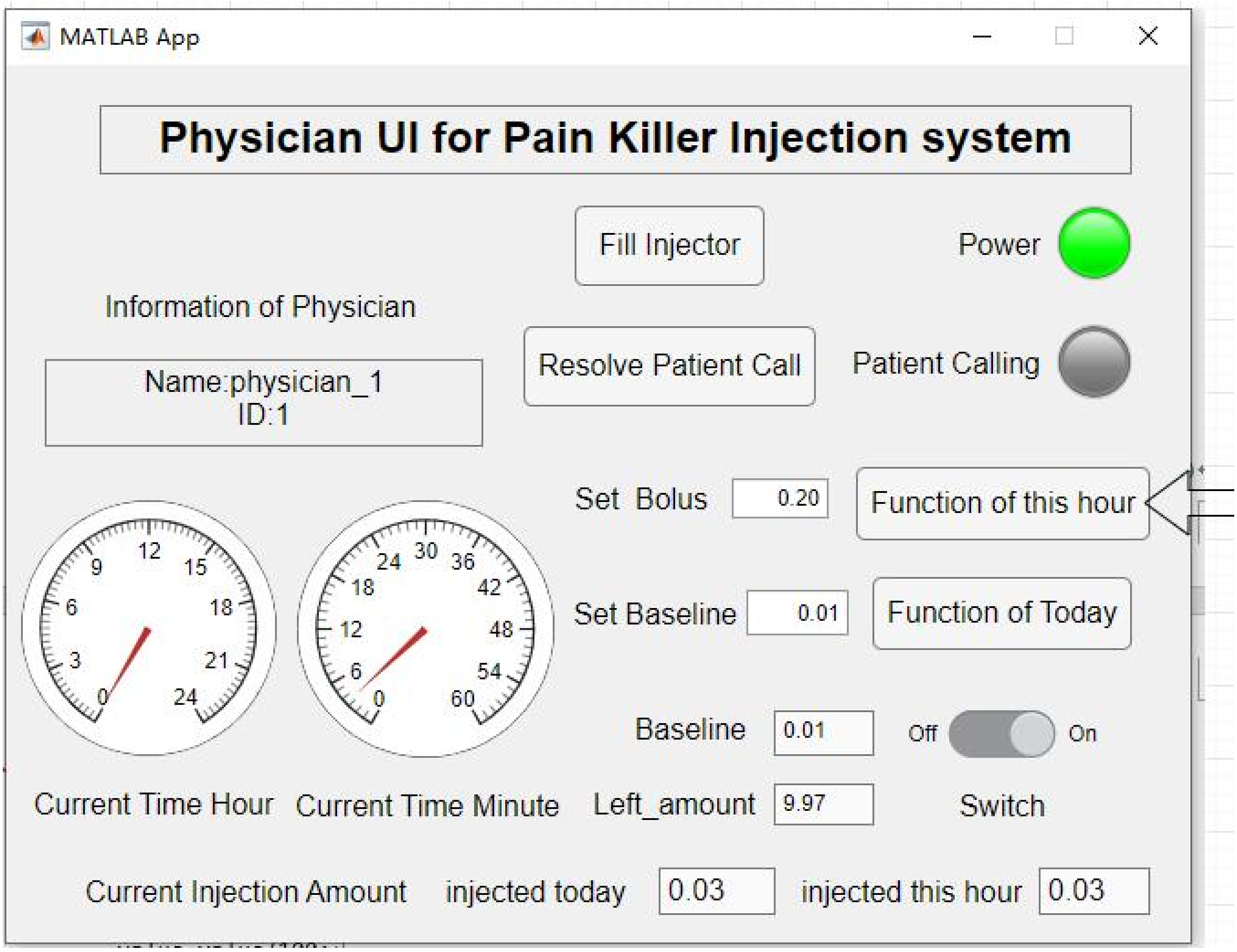
injected this hour 0.03

Code:

```
function ResolvePatientCallButtonPushed(app, event)
    % reslove call from patient
    app.resolveCall;
end
```

S3.7 Show function of inject amount and time during this hour

Physician can see the plot whose x-axis is time and y-axis is inject amount of this hour.



Name:physician_1
ID:1

Set Bolus 0.20

Set Baseline 0.01

Baseline 0.01

Left_amount 9.97

injected today 0.03

injected this hour 0.03

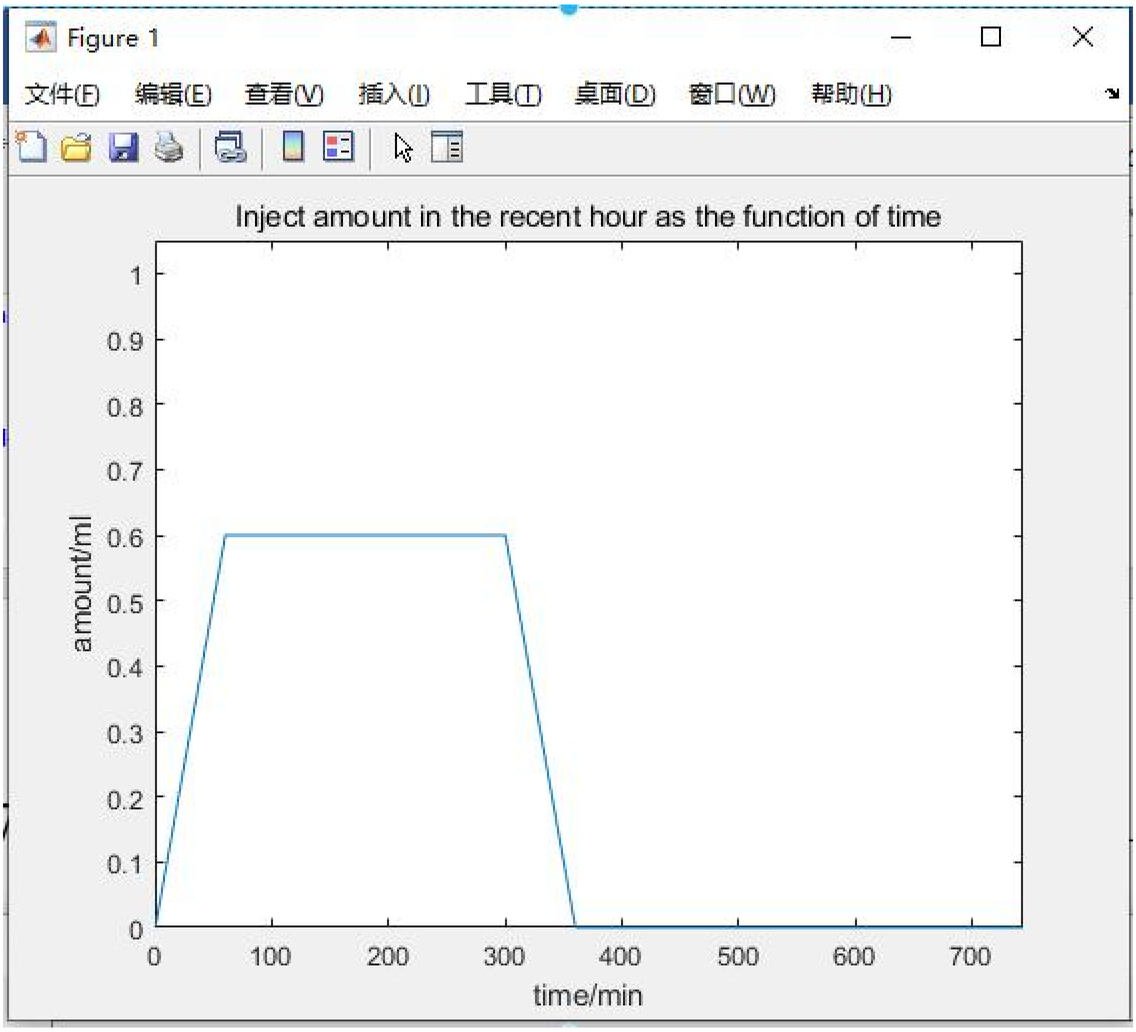


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Inject amount in the recent hour as the function of time
amount/ml
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
600
700
time/min

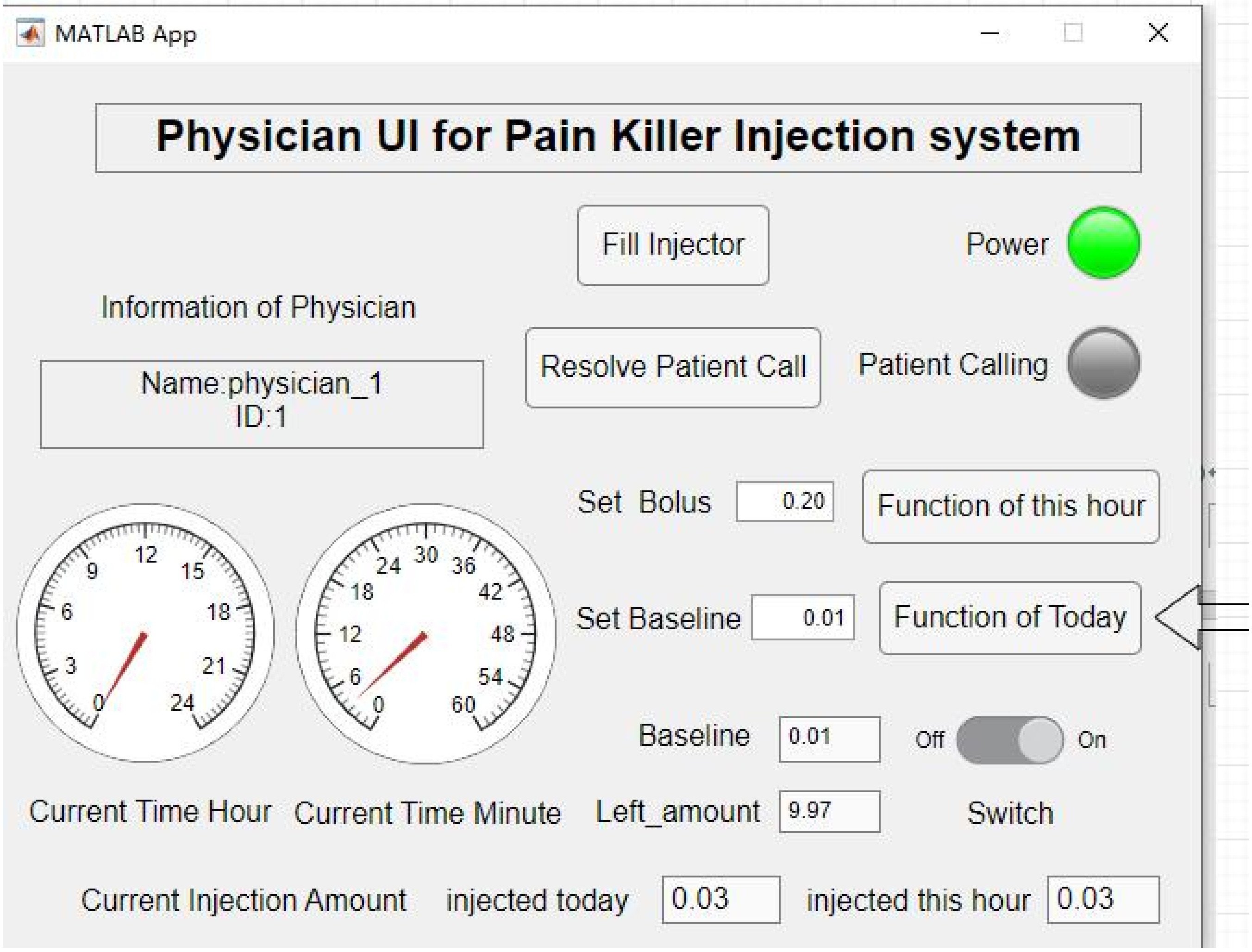
Code:

```
function FunctionofthishourButtonPushed(app, event)
    func_hour(app);
end


function func_hour(app)
    app.hour_figure = figure;
    x=0.1:0.1:1440;
    y=app.controller.func_hour_vec;
    plot(x,y);
    xlabel('time/min');
    ylabel('amount/ml');
    title('Inject amount in the recent hour as the function of time')
    xlim([0,app.controller.minute_elapsed]);
    ylim([0,1.05]);
end
```

S3.8 Show function of inject amount and time during this day

Physician can see the plot whose x-axis is time and y-axis is inject amount of this day.



MATLAB App

Physician UI for Pain Killer Injection system

Information of Physician

Name:physician_1
ID:1

Set Bolus 0.20

Set Baseline 0.01

Baseline 0.01

Left_amount 9.97

Off On Switch

Current Injection Amount injected today 0.03 injected this hour 0.03

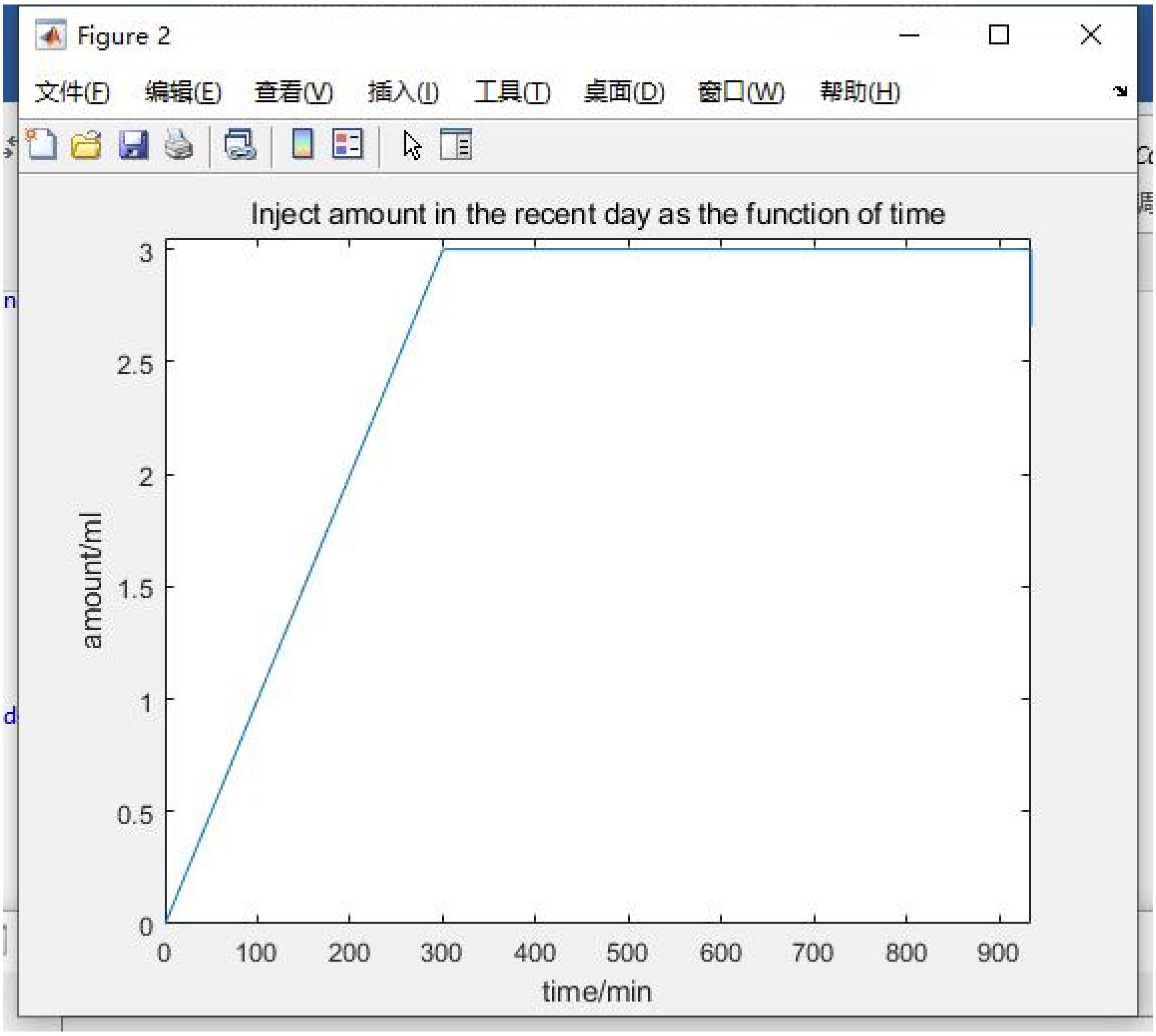


Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Inject amount in the recent day as the function of time
3
2.5
2
1.5
1
0.5
0
amount/ml
0
100
200
300
400
500
600
700
800
900
time/min

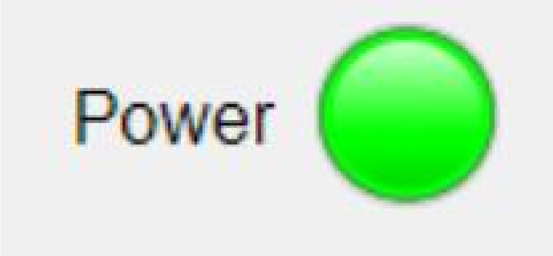
Code:

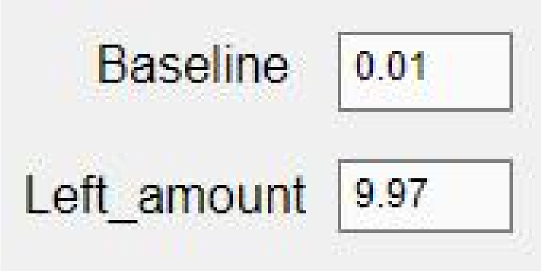
```
function FunctionofTodayButtonPushed(app, event)
    func_day(app);
end


function func_day(app)
    app.day_figure = figure;
    x=0.1:0.1:1440;
    y=app.controller.func_day_vec;
    plot(x,y);
    xlabel('time/min');
    ylabel('amount/ml');
    title('Inject amount in the recent day as the function of time')
    xlim([0,app.controller.minute_elapsed]);
    ylim([0,3.05]);
end
```

S3.9 Update power lamps, baseline and left_amount.

Every time the TimerFnc of t is called, the changeState will be called, so the power lamps, baseline and left_amount will be updated.




Baseline 0.01

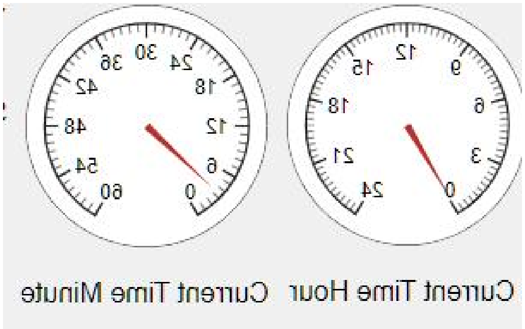
Left_amount 9.97

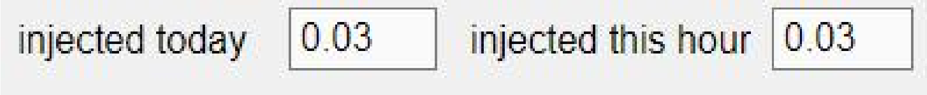
Code:

```
function changeState(app,state,Baseline,Left_amount)
    switch state
        case 0
            app.PowerLamp.Color=[0.5,0.5,0.5];
        case 1
            app.PowerLamp.Color=[0,1,0];
    end
    bas=sprintf('%.2f',Baseline);
    app.BaselineTextArea.Value=bas;
    left=sprintf('%.2f',Left_amount);
    app.Left_amountTextArea.Value=left;
end
```

S3.10 Update current time

Every time the TimerFnc of t1 is called, the showTime will be called, so the current time, day_injected and hour_injected will be updated.




injected today 0.03

injected this hour 0.03

Code:

```
        % print hour,minute,inject this day,inject this hour
        function showTime(app,Hour,Minute,Day_injected,Hour_injected)
            app.CurrentTimeHourGauge.Value=Hour+Minute/60;
            app.CurrentTimeMinuteGauge.Value=Minute;
            day_inj=sprintf('%.2f',Day_injected);
            app.injectedtodayTextArea.Value=day_inj;
            hou_inj=sprintf('%.2f',Hour_injected);
            app.injectedthishourTextArea.Value=hou_inj;
            %draw_func(app);
        end
```

Starter implementation

Starter is implemented in main.m. It is used to clear previous unnecessary things and create new object to launch the program.

Code:

```
close all force;
clear;
clc;

if ~isempty(timerfind)
    stop(timerfind);
    delete(timerfind);
end


control = controller;
patientapp = patient_UI;
physicianapp = physician_UI;

control.patient_UI=patientapp;
patientapp.controller=control;
control.physician_UI=physicianapp;
physicianapp.controller=control;
```