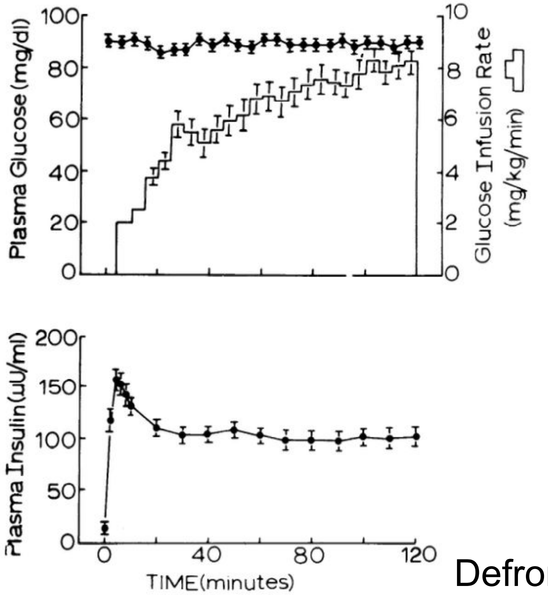
Algorithm for the Infusion Rate of Glucose During an Insulin Clamp

Jason Blohm, Nicholas Diehl, Joe Jeffrey, Sheng-Yau Lim
GlucoReg

Background: Hyperinsulinemic clamp

- Plasma insulin rate is raised and held at 120 μU/mL via constant insulin infusion
- Glucose is infused and blood glucose levels are measured; GIR adjusted
- Measures <u>insulin sensitivity</u> in patient
- Patients: Pre-diabetics, diabetics, those with endocrine & metabolic disorders



Defronzo et al., 1979

Refresher: Problem Statement

- In hyperinsulinemic clamp studies, Dr. Luther adjusts GIR on the fly based on his clinical judgment
- This can lead to inaccurate adjustments which can affect subject safety and data validity
- Some people claim that an algorithmic approach works, but no one Dr. Luther has talked to has been successful
- We will develop an algorithm that allows researchers to perform these studies in a more controlled manner

Refresher: Needs Assessment → Provider

Interface

1. Should be simple to understand
2. Should include inputs for all possible variables the physician may want to change: target glucose level, insulin clamp level, demographic data, time of experiment
3. Given patient demographics and history, should simulate the glucose level over time, prior to clinical testing

Refresher: Needs Assessment → Provider

Algorithm

1. Should calculate the amount of glucose uptake based on the constant insulin infusion rate the physician specifies
2. Depends on actual glucose infusion rate (not suggested)
3. Should output a recommended glucose infusion rate that accounts for the time delay in measuring glucose level from blood sample (t-1)

Refresher: Needs Assessment → Provider

Timing

1. Runtime -- should provide physician with proper glucose infusion rate (GIR) within 10 seconds of inputting the current glucose level
2. Should include an easy to navigate UI for immediate data entry

Refresher: Needs Assessment → Patient

Safety

1. Ensure that glucose levels do not exceed or drop below safe levels, as determined by the physician
2. Measurements need to be taken every 5 minutes to ensure glucose levels are where they should be. If not, the program should alert the physician (future iteration)
3. Must run smoothly so that no bugs interrupt the program

Refresher: Needs Assessment → System

Applicability and cost

1. Should be applicable to different physicians and different hospitals performing the same studies
2. Should be open source
3. Results from these studies should lower healthcare costs in the future

Progress: Multiple Regression Analysis

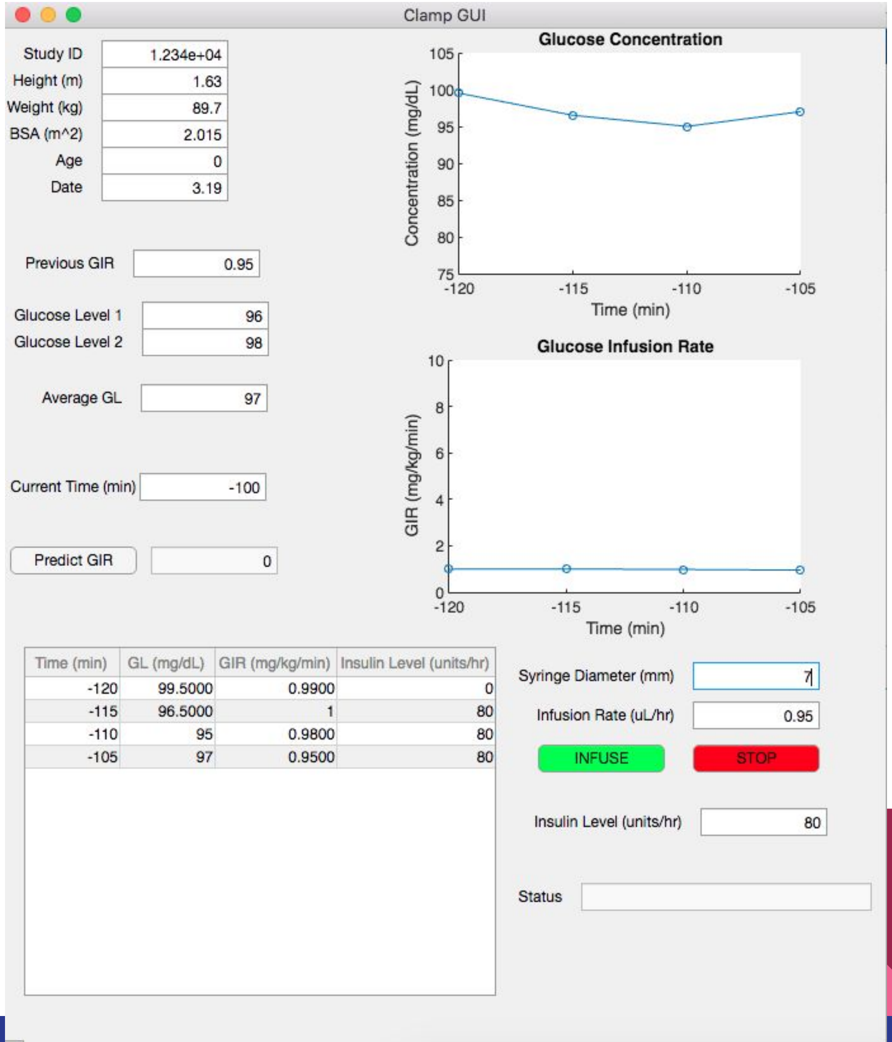
```
Linear regression model:
    GIR ~ 1 + Height + Weight + BSA + Age + Gender + Race
```

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	3.2236	2.011	1.603	0.11325
Height	15.35	5.7597	2.6651	0.0094679
Weight	0.19485	0.083456	2.3348	0.022309
BSA	-20.895	8.0997	-2.5797	0.0119
Age	-0.014226	0.0054871	-2.5927	0.011498
Gender	0.14254	0.16286	0.87521	0.38433
Race	-0.32696	0.22619	-1.4455	0.15261

Progress: GUI

- Will take inputs and generate a predicted GIR
- Will control a syringe pump
- Will output data to an excel file



Code: Algorithm

```
time = cell2mat(app.UITable.Data(:,1));
gl = cell2mat(app.UITable.Data(:,2));
gir = cell2mat(app.UITable.Data(:,3));
%               x1 = linspace(-120,120,1000);

% Plot the GIR and GL

plot(app.glucoseinfus, time, gir,'-o');
plot(app.glucoseconc, time, gl,'-o');


    app.glucoseinfus.YLim = [0 10];
    app.glucoseconc.YLim = [75 105];

% Update the time
app.CurrentTime.Value = curtime + 5;

% Update the Infusion Rate
app.InfusionRate.Value = prevgir;
```

```
% Button pushed function: Button
function ButtonButtonPushed(app, event)

    % Update number of button pushes
    app.buttonpush.Value = app.buttonpush.Value +1;
     butpush = app.buttonpush.Value;

     % initialize variables
   studyid = app.StudyID.Value;
    height = app.Height.Value;
    bsa = app.BSA.Value;
    weight = app.Weight.Value;
    age = app.Age.Value;
    prevgir = app.PrevGIR.Value;
    gl_one = app.GlucoseLevel1.Value;
    gl_two = app.GlucoseLevel2.Value;
    avg_gl = mean([app.GlucoseLevel1.Value, app.GlucoseLevel2.Value]);
    app.AverageGL.Value = avg_gl;
    curtime = app.CurrentTime.Value;
    LD = app.LowDose.Value;


    % Calculate the future GIR
    if prevgir==0 && curgl==0
        app.PredictGIR.Value = 1+ 15.35*height + .19485*weight - 20.895*bsa - age*.014226;
    else
        % This is where we will put in the algorithm
        app.PredictGIR.Value = 0;
    end
    % Update table
    app.UITable.Data{butpush,1} = curtime;
    app.UITable.Data{butpush,2} = avg_gl;
    app.UITable.Data{butpush,3} = prevgir;
    if LD == 1
        app.UITable.Data{butpush,4} = 'Low Dose';
    else
        app.UITable.Data{butpush,4} = 'High Dose';
    end
```

Progress: Machine Learning

Support Vector Regression Kernels to build algorithm using scikit in Python

- Open Source
- One of the best coding languages for Machine Learning
- y0 as a function of demographics (initial infusion rate from multiple regression)
- Δy as a function ONLY of Δx — linearity, one independent variable
- cross validation

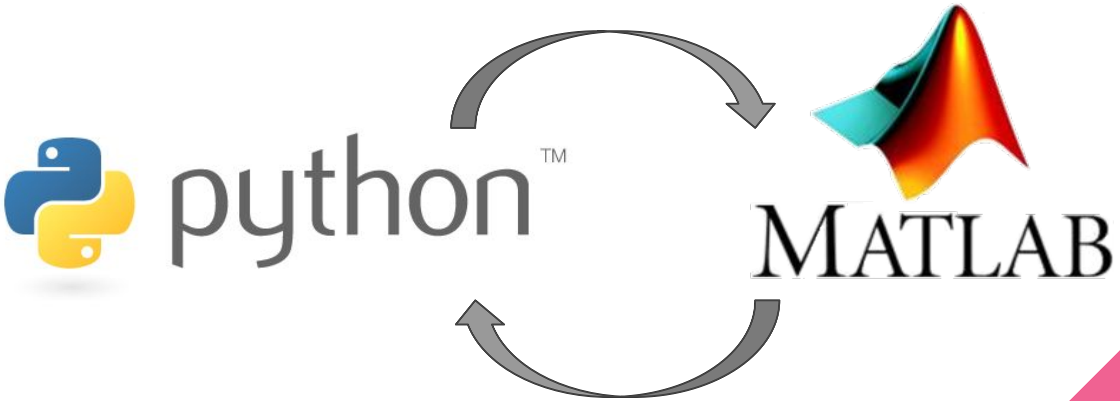
Code: Python Algorithm

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn import svm

# Read Patient File
patientdata = pandas.read_excel('7832002.xlsx', sheet_name='DeltaData')
from sklearn.linear_model import LinearRegression

# Randomly Divide data into training and cross validation set
# (test_size is percentage of data used to evaluate)
X = patientdata.drop('delta glucose',axis = 1)
Y = patientdata.drop('delta gir',axis = 1)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.33, random_state = 5)

# Build Linear Regression Model
lm = LinearRegression()
lm.fit(X_train,Y_train)
pred_train = lm.predict(X_train)
pred_test = lm.predict(X_test)

print('Estimated intercept coefficient:', lm.intercept_)
```

```
# Cross Validation
from sklearn.model_selection import cross_val_score
clf = svm.SVC(kernel='linear', C=1)
scores = cross_val_score(clf, X, Y, cv=5)
scores
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

# Scatterplot
plt.scatter(X,Y)
plt.xlabel("Change in GIR (ml/h)")
plt.ylabel("Change in Glucose (mg/dl)")
plt.title("Change in GIR vs Change in Glucose")
plt.show()

# Predict Delta Glucose level from Delta GIR
lm.predict(X)

# Plot true vs predicted Delta Glucose level
plt.scatter(Y, lm.predict(X))
plt.xlabel("Actual Delta GL")
plt.ylabel("Predicted Delta GL")
plt.title("Actual vs Predicted Delta GL")
```

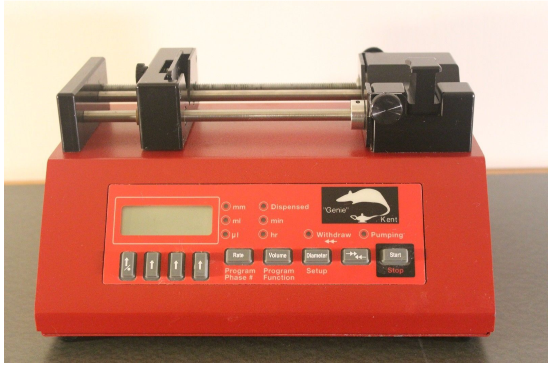
Code: Next Steps

- Enable Python script to accept new data during study to improve algorithm
- Implement code to communicate predicted GIR from Python to Matlab
- Consider including Demographic Parameters to improve algorithm accuracy



Pump POC

- Syringe Pump POC
  - Kent Scientific Genie Syringe Pump
  - Glucose solution in syringe
  - Communicate from MATLAB to pump via serial communication
  - Monitor physiological signals
  - Monitor pump controls



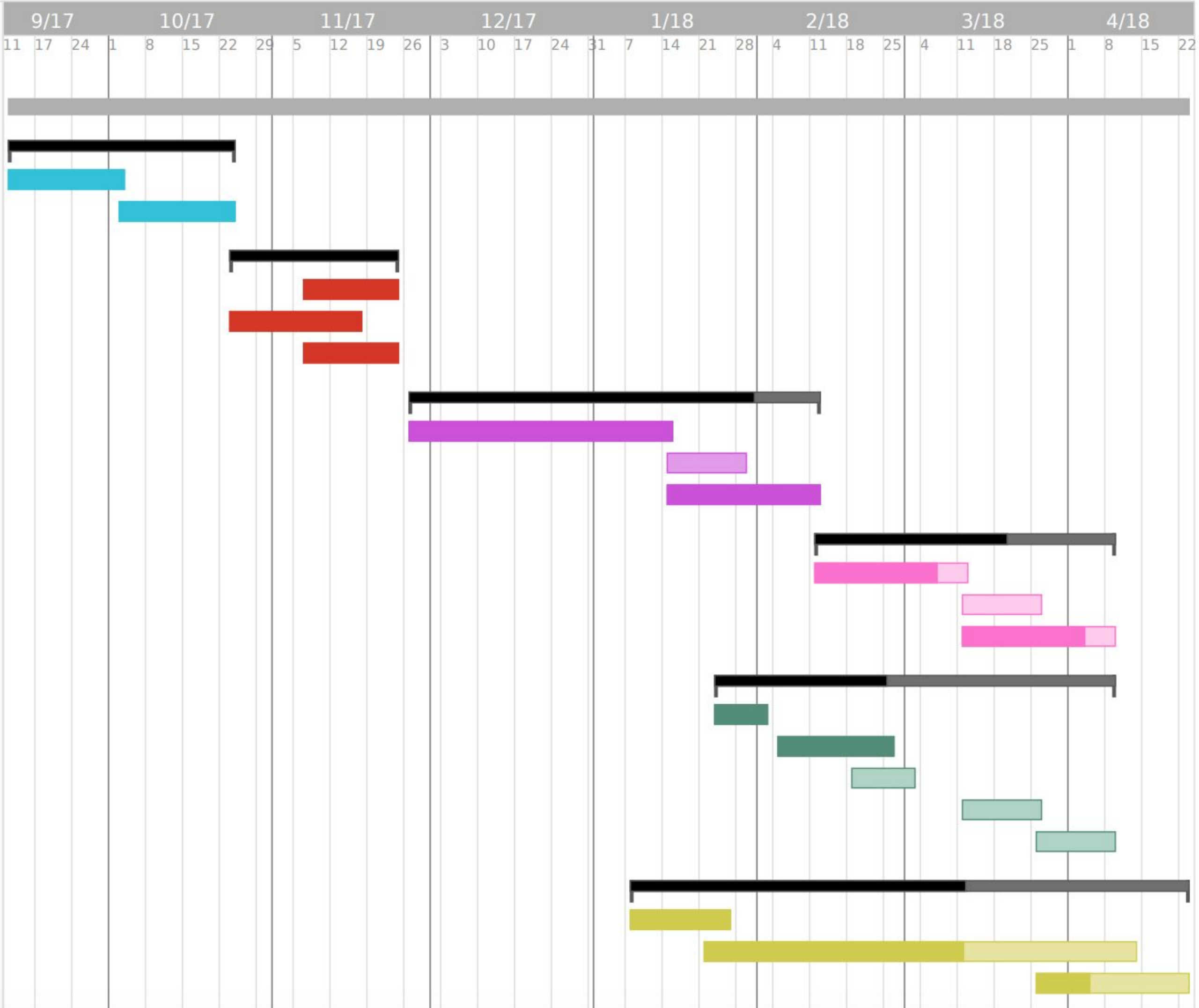
Code: Pump

```
% settings serial port
% clear all instruments
tmp=instrfind();
delete(tmp);


% settings serial port


app.serialPump=serial('/dev/tty.usbserial','BaudRate', 1200, 'DataBits',8);
set(app.serialPump,'Terminator','CR','TIMEOUT',0.5);


% open serial port
 try
    fopen(app.serialPump);
catch
    msgbox('Could not open serial port');
end
```

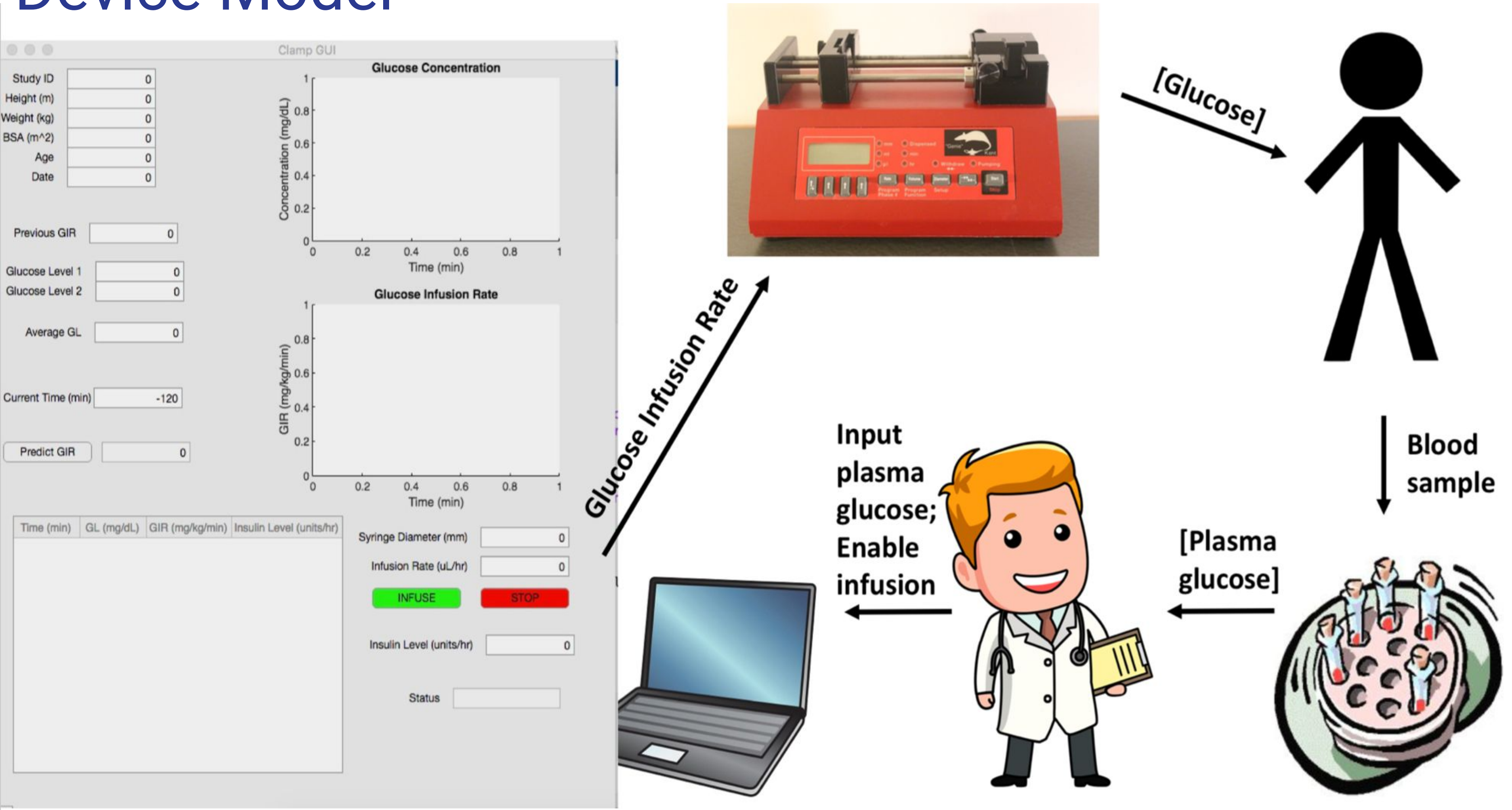
```
% Value changed function: Infuse
function InfuseValueChanged(app, event)
    % send command
    diam = app.dia.Value;
    rate = app.InfusionRate.Value;

    % Confirmation
   promptMessage = sprintf('Are you sure you would like to infuse?');
   button = questdlg(promptMessage, '!! Warning !!', 'Continue', 'Cancel', '
   if strcmpi(button, 'Cancel')
               return; % Or break or continue
   elseif strcmpi(button, 'Continue')
       fprintf(app.serialPump, 'DIA %5.3f\n',diam);
       fprintf(app.serialPump, 'DIA\n');
       pause(.5)
       fprintf(app.serialPump, 'RAT UH');
       pause(1);
       fprintf(app.serialPump, 'RAT %5.3d\n',rate);
%                 fprintf(app.serialPump, 'RAT\n');
       pause(.5);
       fprintf(app.serialPump, 'RUN');
       warning('off')
       app.Status.Value = fscanf(app.serialPump);        % Checks if commanc
   else
       return
   end
```

Gantt Chart

Glucose Regulation Algorit...	**72%**
Preliminary Brainstorming	**100%**
Familiarization and meeting Dr. Luther	100%
Brainstorming	100%
Preliminary Data Analysis	**100%**
Data storage and sorting	100%
Database and demographic analysis	100%
Multiple regression analysis	100%
Algorithm - first round	**84%**
Development of algorithm - first draft	100%
Clinical testing (1)	0%
Algorithm iterations (1)	100%
Algorithm - second round	**64%**
Implementation of machine learning	80%
Clinical testing (2)	0%
Algorithm iterations (2)	80%
Pump interface	**43%**
Familiarization with pump/Brainstorm...	100%
Prototype	100%
First round of testing	0%
Prototype modification	0%
Finalization	0%
Algorithm + Pump Finalization	**60%**
Creation of user interface	100%
Finalization of algorithm, UI, and pum...	60%
Design day preparation	35%



Device Model



BMES Grant Application

- Finalized aside from video of prototype and proof of design
- Will need a letter of support
- Just emailed you what we have

Algorithm for the regulation of glucose infusion

Jason Blohm (B.E., 2018)
Nicholas Diehl (B.E., 2018)
Joseph Jeffrey (B.E., 2018)
Sheng-Yau Lim (B.E., 2018)

Vanderbilt University

Faculty Advisor: Dr. Matthew Walker
Department Chair: Dr. Philippe Fauchet

Questions?