Dealing with Diabetes: The Road to Developing an Artificial Pancreas

https://www.sciencebuddies.org/science-fair-projects/project-ideas/HumBio_p040/human-biology-health/developing-an-artificial-pancreas
(https://www.sciencebuddies.org/science-fair-projects/project-ideas/HumBio_p040/human-biology-health/developing-an-artificial-pancreas)

Procedure PDF Date: 2023-10-25

Experimental Procedure

Note: This engineering project is best described by the **engineering design process,** as opposed to the **scientific method.** You might want to ask your teacher whether it's acceptable to follow the engineering design process for your project before you begin. You can learn more about the engineering design process in the Science Buddies Engineering Design Process Guide (http://www.sciencebuddies.org/science-fair-projects/engineering-design-process/engineering-design-process-steps).

1. Build your conductivity sensor using two strips of aluminum foil and a piece (or pieces) of floating material. There are several options to build one (Figure 4). Make sure the sensor fits in the container you will use for water (Figure 5).
    a. The exact dimensions of the sensor do not matter. The general idea is that it should have two strips of aluminum foil, spaced apart by several inches, so they will sit at opposite ends of your water container.
    b. The sensor should float so that the amount of submerged surface area on the aluminum foil stays constant while you add water (the amount of submerged surface area will affect the conductivity reading).
    c. You will need to grab on to each piece of aluminum foil with an alligator clip, so make sure you leave part of each piece sticking up.

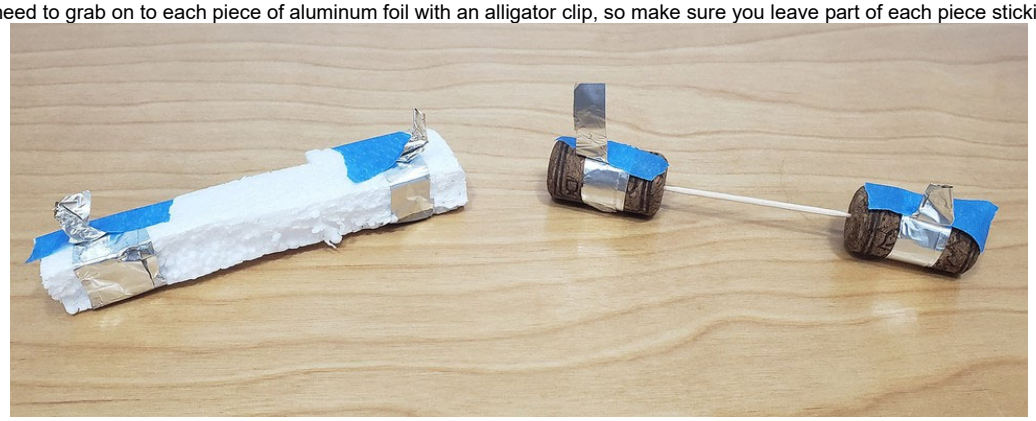

Figure 4. Two options for building a conductivity sensor. Left: a piece of packing foam with two strips of aluminum foil wrapped around its ends. Right: two corks with aluminum foil strips wrapped around them, spaced apart by a toothpick.

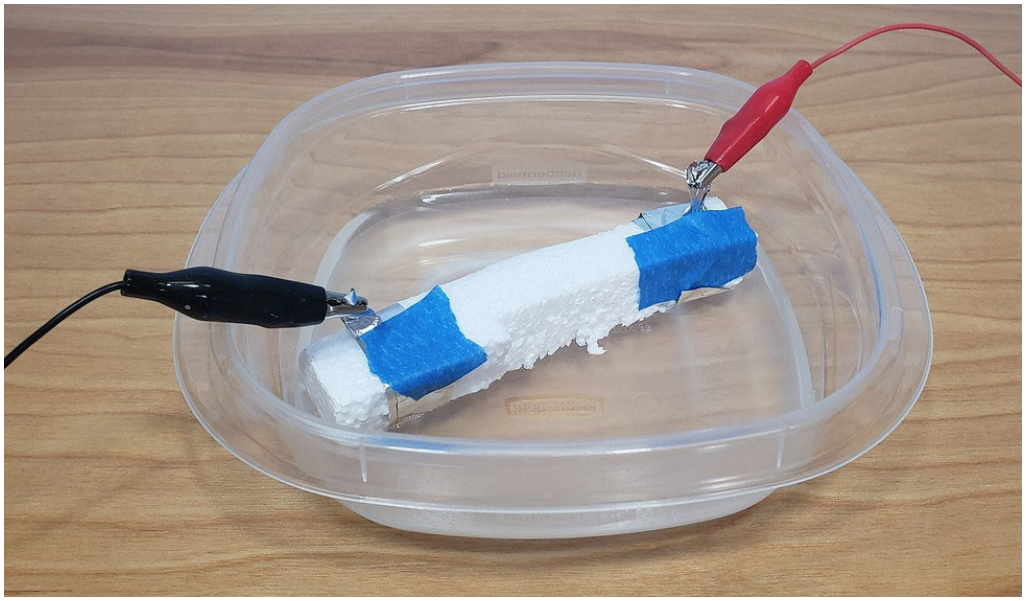

Figure 5. Conductivity sensor floating in a container of water with alligator clips attached to the aluminum foil.

2. Build the circuit. Figure 6 shows a "breadboard view" diagram. If you prefer to work with the circuit diagram, refer to Figure 3 in the introduction. **Do not connect your Arduino to the USB cable yet.** This will ensure that your circuit remains powered-down while you build it.
    a. Connect 5V on your Arduino to the power (+) bus on your breadboard.
    b. Connect the left and right power buses to each other.
    c. Connect GND on the Arduino to the ground (-) bus on your breadboard.

d. Connect the left and right ground buses to each other.
e. Put the MOSFET in the breadboard with the large metal tab facing to the left, and the writing on the front facing to the right. Each pin on the MOSFET should be in a different row of the breadboard.
f. Connect the MOSFET's first pin on the left (when the writing is facing you, so the bottom-most pin in Figure 6) to Arduino digital pin 11.
g. Connect the MOSFET's third pin to the ground bus.
h. Put the 100 kΩ resistor into two different rows on the breadboard. Connect one end of the resistor to the power bus. Connect the other end to Arduino analog input pin A0.
i. Use jumper wires and alligator clips to connect your conductivity sensor. Connect one end of the sensor to the same end of the resistor that is connected to pin A0. Connect the other end to the ground bus.
j. Use jumper wires and alligator clips to connect the pump. Connect the tab labeled with a "+" to the power bus. Connect the other tab to the middle pin of the MOSFET.

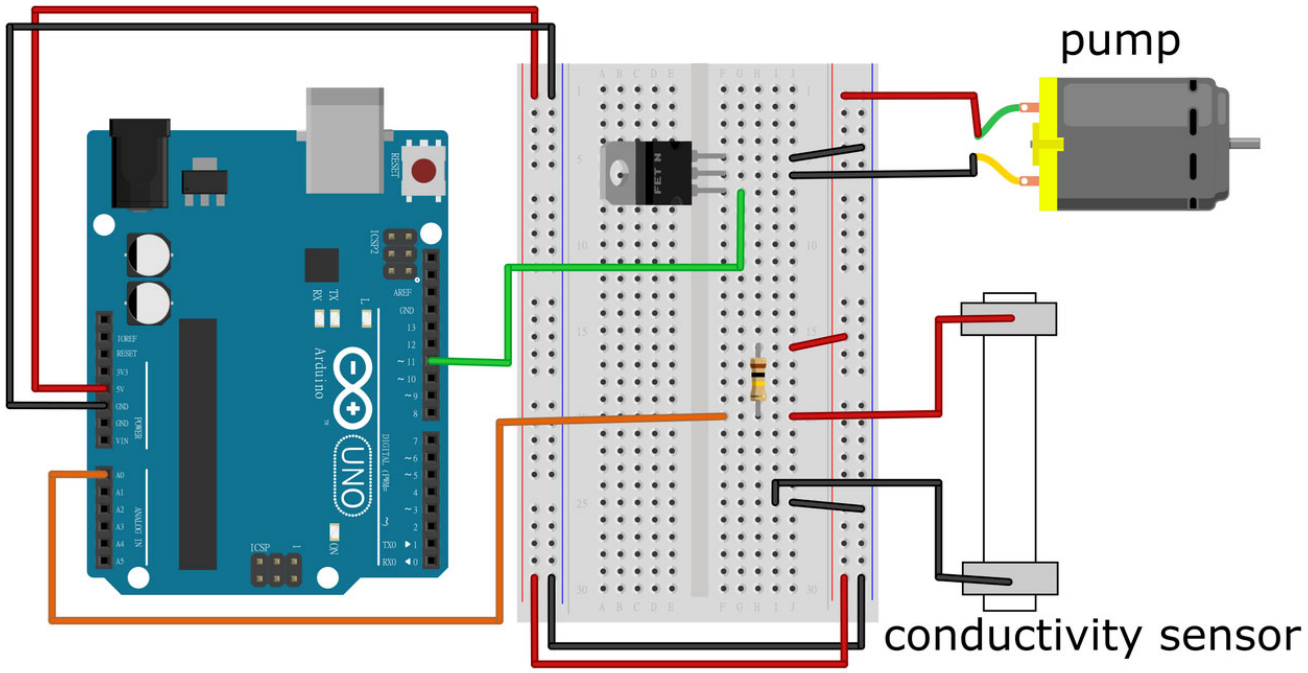

Figure 6. A breadboard diagram of the artificial pancreas model circuit. Click here to open a larger version of the diagram. (https://www.sciencebuddies.org/Files/7660/18/artificial-pancreas-breadboard-diagram.png)

3. Before you continue, you will need to calibrate your conductivity sensor. Fill one of your containers about halfway with distilled water. Place your conductivity sensor in the container. Make sure it floats. You may need to adjust the alligator clips.
4. **Temporarily disconnect one alligator clip from the pump so it does not turn on during the calibration process.**
5. Plug one end of the USB cable into your Arduino and the other end into your computer. This will power up the Arduino and provide power to your circuit. If you have previously uploaded a different program to the Arduino, that program will start running automatically, but that is OK for now (which is why you disconnected your pump, just in case the Arduino code turns on pin 11). There are two different ways to take the calibration reading. Using a multimeter is not required, but it can be more convenient since you do not need to use a computer.
    a. If you have a multimeter available, set it to measure DC voltage and connect it to the circuit as show in Figure 7.
        i. Connect the multimeter's black probe to the ground bus and connect the red probe to the breadboard row that is connected to Arduino pin A0.
        ii. Record the voltage reading. Convert it to an equivalent analogRead value by dividing it by 5, then multiplying by 1023 and rounding (for example, if your reading is 4 volts, the value would be 4/5×1023 = 818.4, which rounds to 818).
    b. If you do not have a multimeter available, download the calibration code (http://www.sciencebuddies.org/cdn/Files/18825/5/pump_calibration_code.ino) to your computer. Then upload it to your Arduino. In the Arduino IDE, select Tools→Serial Monitor. The code will print out the analogRead value from the conductivity sensor. The value might fluctuate slightly. This is OK. Record the average value over a period of a few seconds.

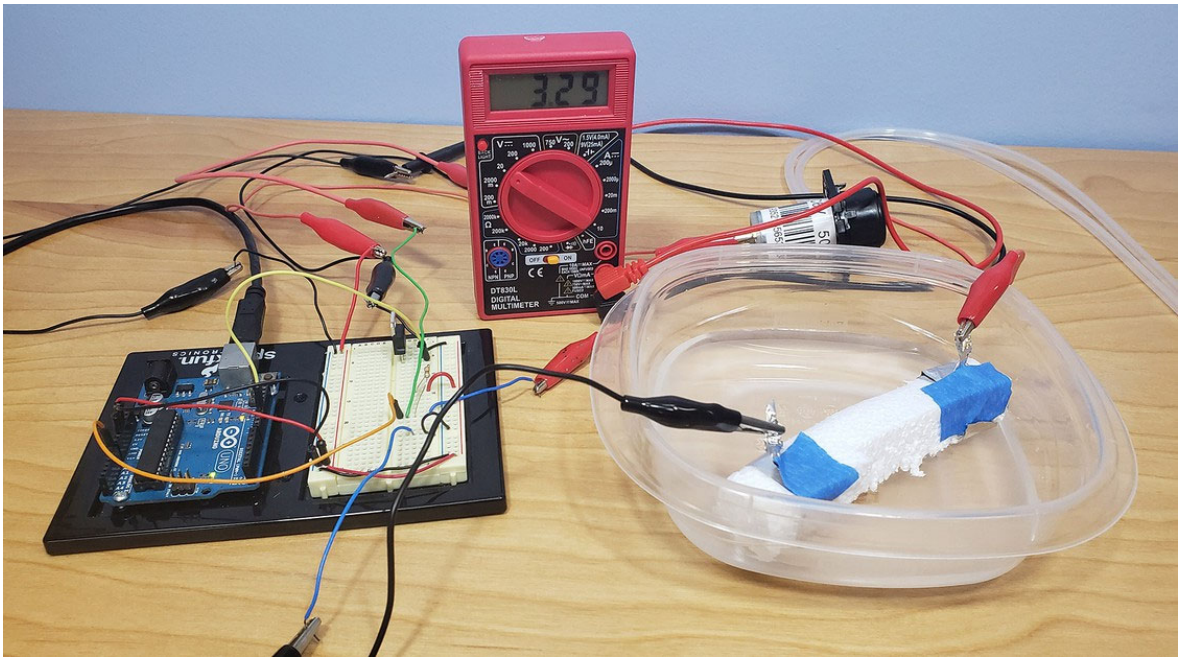

Figure 7. Multimeter connected to measure voltage from the conductivity sensor.

6. Fill your other container about halfway with tap water. Optionally, add a few drops of food coloring to make it easier to distinguish from the distilled water. Remove your conductivity sensor from the container with distilled water, dry it off with a towel, place it in the tap water, and the calibration procedure in step 5.
7. The values you recorded in steps 5 and 6 are the maximum and minimum "blood sugar" values in your model. This can be a voltage (between 0 and 5 volts) or a number between 0 and 1023 (from the analogRead) function. To be clear, your Arduino code will use the number between 0 and 1023, not a number between 0 and 5, so if you measured a voltage, you need to convert it to the 0 1023 range as described in step 5. The "high" value corresponds to pure distilled water, and the "low" value corresponds to pure tap water. When you mix tap water and distilled water, you will get a number somewhere in between those two values. Your goal is to start the conductivity sensor out in a container of pure distilled water (which represents high blood glucose). Your code will measure the voltage from the conductivity sensor and automatically turn on the pump to start adding tap water to the container (to simulate adding insulin). This will cause the voltage to start dropping. When the voltage drops below a certain threshold that you define, the code will automatically turn the pump off. This models the behavior of an artificial pancreas, which would monitor blood glucose levels and apply insulin until glucose levels dropped (or were predicted to drop) to an acceptable level. Remember that your system is a model and responds to changes very quickly (within a few seconds), whereas changes in blood glucose levels can take much longer.
8. Before you continue, you need to make sure you know which way water flows in your pump. Place both ends of the pump's tubing in the container with distilled water. Use alligator clips and jumper wires to connect the pump's electrical tabs directly to 5V (+) and GND (-) on your breadboard (the tab labeled "+" on the pump should go to 5V). This will turn the pump on. One at a time, carefully lift each of the pump's tubes out of the water. Watch closely to observe the direction of water flow (which tube has water dripping out of it?). Use a marker to draw arrows in the direction of water flow for each tube (Figure 8). Note that if you reverse the connections between 5V and GND, water will flow in the opposite direction. When you are done, disconnect the pump's negative wire from the GND pin.
9. Leave the pump's outlet tube in the container of distilled water and place the inlet tube in the container of tap water.
10. Remove the conductivity sensor from the container of tap water, rinse it off with some distilled water, and put it back in the container of distilled water.

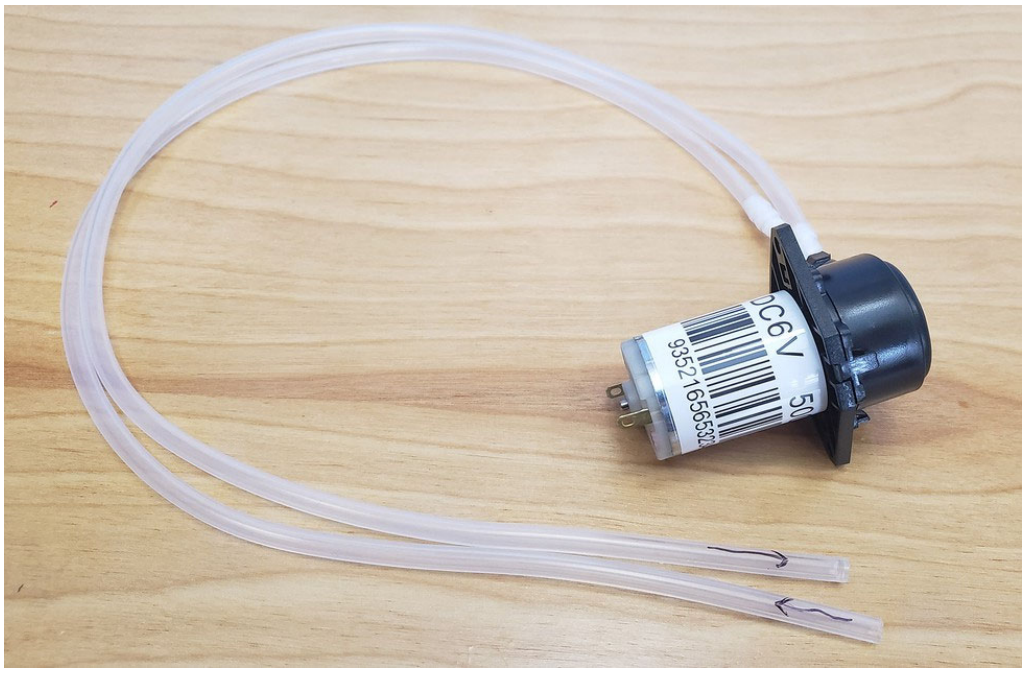

Figure 8. Arrows labeling the direction of water flow in the pump.

11. Download the example code (http://www.sciencebuddies.org/cdn/Files/18826/4/artificial_pancreas_example_code.ino). Read through the commented code and make sure you understand how it works. Choose a value for the "threshold" variable, the point at which you want the pump to turn off. Remember that this should be between the minimum and maximum values you recorded earlier (using the 0–1023 scale, not the 0–5 voltage scale). For example, if you recorded a minimum value of 200 and a maximum value of 800, you could set the threshold for the pump to turn off at 300. Upload the code to your Arduino and make sure you have the serial monitor open.
12. Reconnect the pump's negative wire to the middle pin of the MOSFET on the breadboard. The pump should start right away, pumping tap water into the container with distilled water (Figure 9). As it pumps, **use the pump's outlet tube to gently stir the distilled water container**. This will help evenly distribute the tap water throughout the container. Watch the output on the serial monitor, and/or your multimeter reading if you have it connected.
    a. Does the pump turn off once the value reaches your defined threshold?
    b. Does the value keep changing after the pump turns off?
    c. If your pump does not turn on at all, unplug the USB cable from your Arduino. Double check your wiring, especially the connections to the MOSFET, since it controls the pump.
    d. If your pump never turns off, unplug the USB cable from your Arduino.
        i. Double check your wiring, especially the connections between the resistor, conductivity sensor, and analog input pin.
        ii. Make sure the value you chose for the "threshold" value makes sense, as described in step 9.

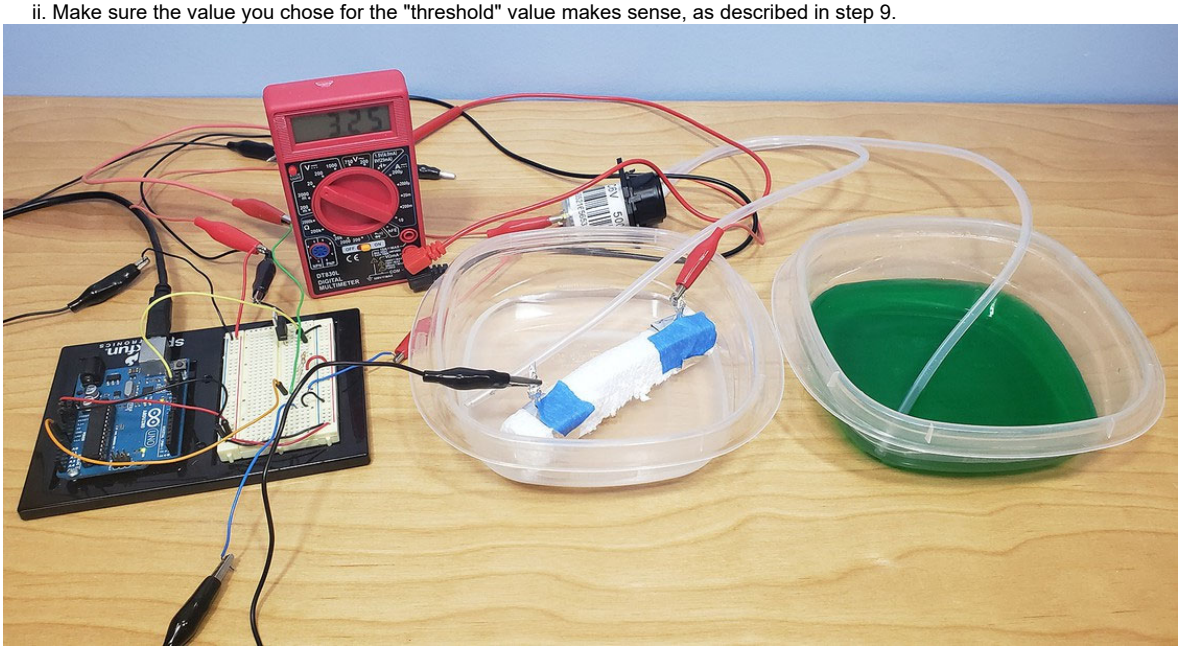

Figure 9. The experimental setup, with the pump arranged to pump tap water into the container of distilled water.

13. Continue to stir the water for a minute or so after the pump turns off. Once you are confident the pump is staying off, disconnect the USB cable from your Arduino. This will stop it from printing data to the serial monitor.
14. (Note: the instructions in this step are for a Windows computer. You will need to figure out how to do it if you are using another operating system). Click in the serial monitor and press CTRL+A to select all of the data. Press CTRL+C to copy it, and then open a .txt file in a program like Notepad and paste the data in. Save this file.
15. You will need to import your data into a spreadsheet program like Microsoft Excel® or Google Sheets®. Most spreadsheet programs can import text files and split the data into different columns. For example, Excel can import text files using a "delimiter," a special character (like a tab or a space), to indicate a new column. Import your data so you have the time in one column and the analog reading in a separate column.
16. Make a graph of your sensor reading (y axis) vs time (x axis).
    a. How quickly does your sensor reading reach the threshold value after the pump initially turns on?
    b. Does the sensor reading overshoot, or go past, the threshold value?
    c. Where does the sensor reading eventually stabilize?
17. Now it is time to apply the engineering design process and **iterate** to improve your model. Look at the Arduino code and think of ways to improve it. How can you get to the threshold value quickly without overshooting too much? This is an open-ended problem. Here are some suggestions to get you started.
    a. You can control the pump's speed using the **analogWrite** command. analogWrite accepts a value between 0–255 that corresponds to the pump's speed (e.g., 0 is off, 127 is about half-speed, and 255 is full-speed). What happens if you run the pump at a lower speed?
    b. You could try a different type of feedback control instead of basic on/off control. For example, instead of running it a constant speed, a **proportional controller** changes the pump's speed based on the measurement **error**, or the difference between the target sensor reading and the actual sensor reading. In other words, when the error is large (the actual reading is very far away from the target reading), the pump will run faster. As the error gets smaller, the pump will start to slow down.
    c. Advanced students can look up other types of control like **proportional-integral-derivative (PID)** control, which involves more advanced mathematical functions and calculus.
18. Try editing the Arduino code, re-uploading it, and testing your model again. In between tests, rinse and dry your containers and conductivity sensor. You should start out with fresh distilled water each time.
19. Compare your new graph of voltage vs. time to your previous graph. It may help to plot them on the same axes, with the x-values shifted if needed so the points where the pump starts align with each other.
20. Remember that your system is a *model* of an artificial pancreas, where the sensor reading (voltage) represents blood glucose levels. Think back to the information about diabetes in the introduction. What would you need to consider when designing a real artificial pancreas? What kind of response (the shape of the voltage vs time graph) would be ideal for a person with diabetes? What are the differences in how insulin works (and how long it takes) compared to your model? What challenges do these differences create? Your model turns insulin on and off, but what happens if blood glucose levels end up too low (overshoot) and/or continue to drop? How can an artificial pancreas address this?

Frequently Asked Questions (FAQ)

FAQ for this Project Idea available online at

https://www.sciencebuddies.org/science-fair-projects/project-ideas/HumBio_p040/human-biology-health/developing-an-artificial-pancreas#help.