TASK 3.2C – SIT225

TRUONG KHANG THINH NGUYEN

Setup :

For this task, I selected Ultrasonic Sensor to trigger an alarm whenever the distance it measures exceeds the specified threshold. Specifically, I set two threshold minimum and maximum distances ,respectively. And I created the **Slider** to adjust the maximum threshold and the **Stepper** to modify the Minimum threshold.

In terms of alarming widgets, **LED** and **Status** are used to indicate the triggered alarm. Basically, whenever the measured is less than minimum or exceeds the maximum threshold, both the widgets will be turn on. Finally, I added the Gauge widget to indicate the distance measured for the audience easily keeps track whether the distance exceeds the specified threshold.

Practical Application:

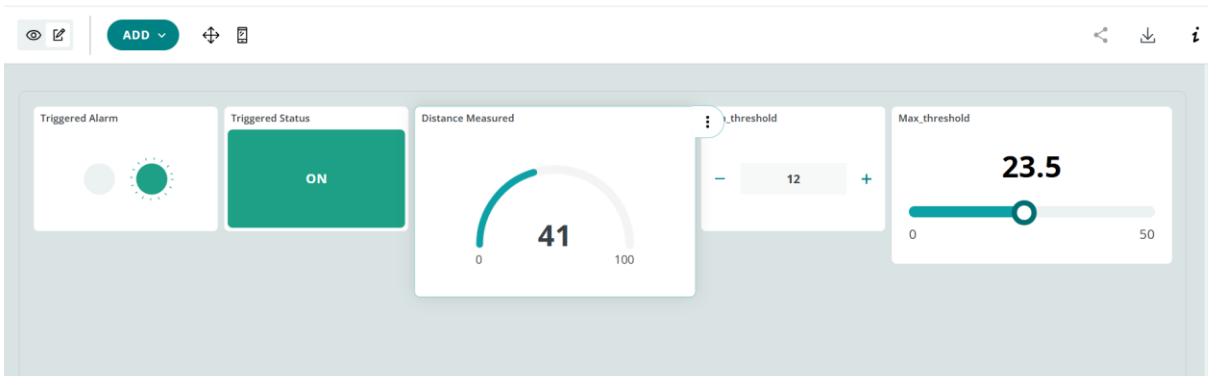
Regarding the minimum threshold, it can be used to prevent vehicles collision whenever

the drivers want to move backwards their cars. On the other hand, with respect to the maximum threshold, it can be utilized into child safety monitoring, For example, if the child wanders more than 50m away from the caregivers it will notify them via a smartphone or wearable devices.

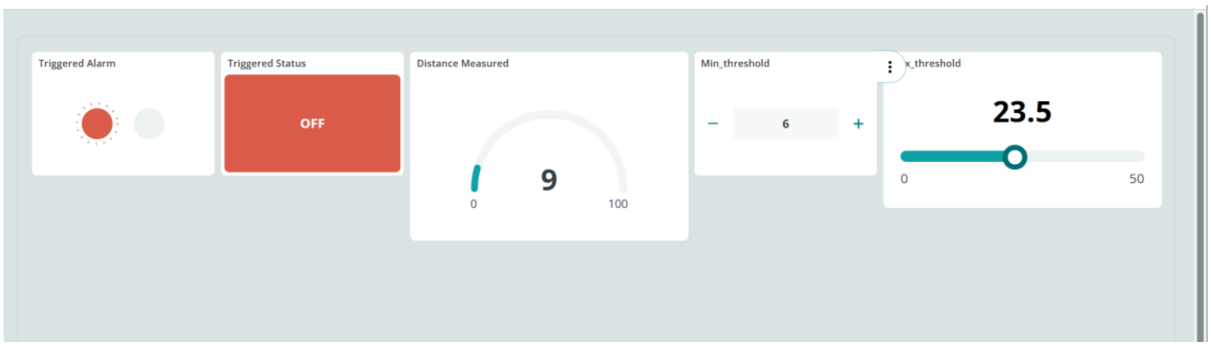
Hypothesis :

As mentioned before, whenever the measure distance exceeds one of the specified threshold. The triggered widgets will be turned on.

Implementation:



As the figure showed, the measured distance was currently is **41**, which exceeds the specified maximum **23.5** so the triggered widgets were being turned on.



However, as I put my hand in front of the Ultrasonic sensor the sensor measured **9** which between the minimum **6** and the maximum threshold **23.5** so the triggered alarm. Therefore, the triggered widgets obviously didn’t turn on.

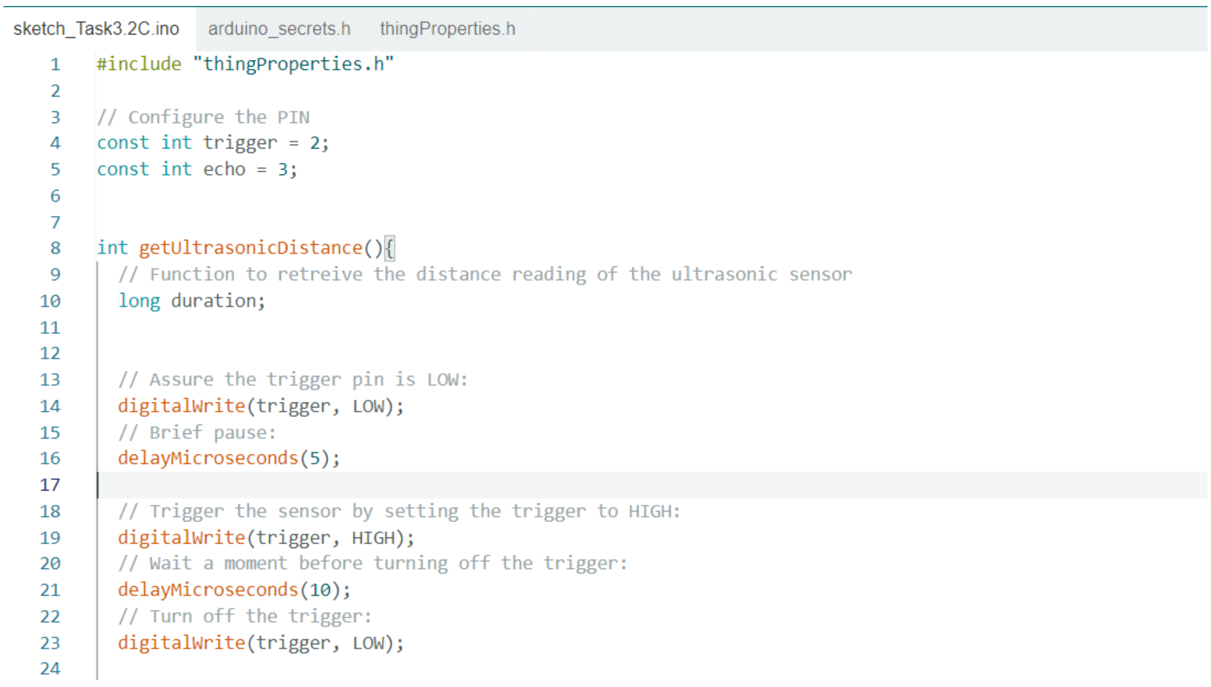
Arduino Code:

Sketch_Task3.2C.ino

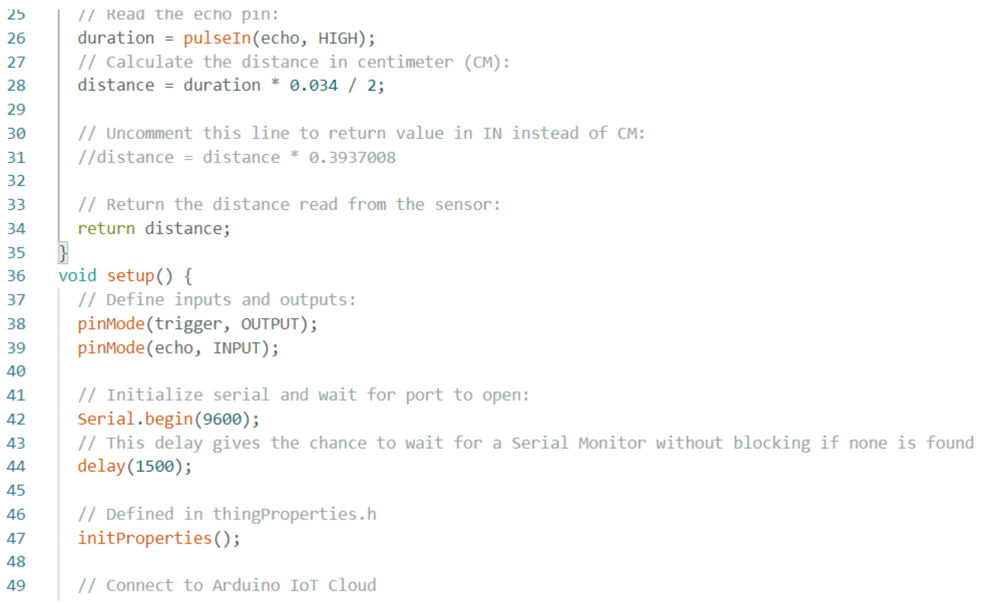
```
#include "thingProperties.h"

// Configure the PIN
const int trigger = 2;
const int echo = 3;

int getUltrasonicDistance(){
 // Function to retreive the distance reading of the ultrasonic sensor
 long duration;

 // Assure the trigger pin is LOW:
 digitalWrite(trigger, LOW);
 // Brief pause:
 delayMicroseconds(5);

 // Trigger the sensor by setting the trigger to HIGH:
 digitalWrite(trigger, HIGH);
 // Wait a moment before turning off the trigger:
 delayMicroseconds(10);
 // Turn off the trigger:
 digitalWrite(trigger, LOW);

 // Read the echo pin:
 duration = pulseIn(echo, HIGH);
 // Calculate the distance in centimeter (CM):
 distance = duration * 0.034 / 2;

 // Uncomment this line to return value in IN instead of CM:
 //distance = distance * 0.3937008

 // Return the distance read from the sensor:
 return distance;
}
void setup() {
 // Define inputs and outputs:
 pinMode(trigger, OUTPUT);
 pinMode(echo, INPUT);

 // Initialize serial and wait for port to open:
 Serial.begin(9600);
 // This delay gives the chance to wait for a Serial Monitor without blocking if none is
found
 delay(1500);

 // Defined in thingProperties.h
 initProperties();

 // Connect to Arduino IoT Cloud
```

```
 ArduinoCloud.begin(ArduinoIoTPreferredConnection);


 /*
   The following function allows you to obtain more information
   related to the state of network and IoT Cloud connection and errors
   the higher number the more granular information you’ll get.
   The default is 0 (only errors).
   Maximum is 4
 */
 setDebugMessageLevel(2);
 ArduinoCloud.printDebugInfo();
}


void loop() {
 ArduinoCloud.update();
 // Your code here
 // Print the distance to the serial monitor:
 distance = getUltrasonicDistance();
 Serial.print("Distance: ");


 if (distance < min_threshold || distance > max_threshold) // If the object is too near the
sensor  or exceeds the maximum threshold, it will trigger the alarm.
 {
  triggered = true;
 }
 else {
  triggered = false;
 }
 Serial.println(distance);
```

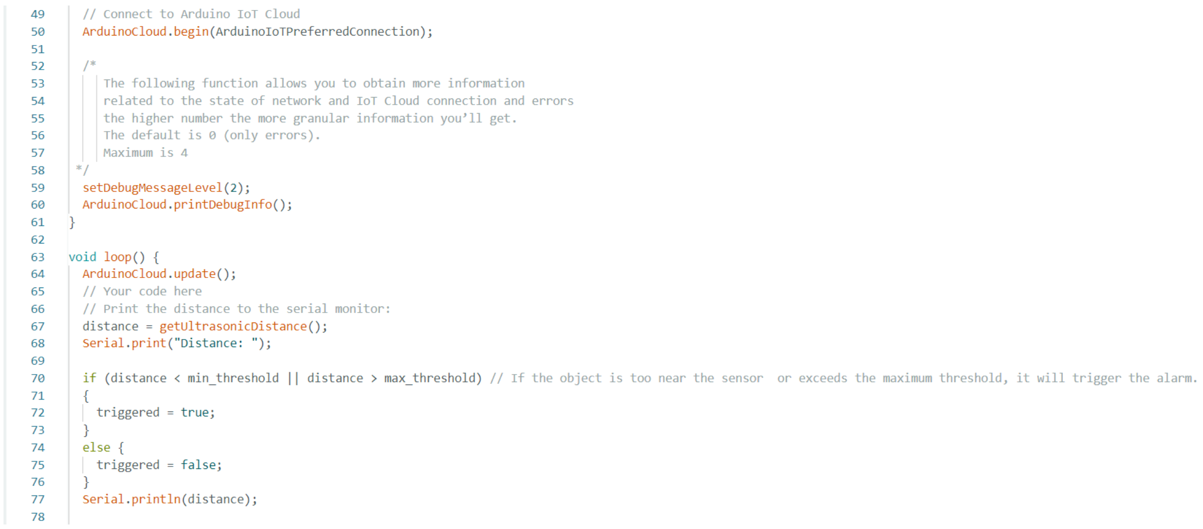
```
  // Wait one second before continuing:
  delay(1000);
}

/*
  Since Distance is READ_WRITE variable, onDistanceChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onDistanceChange()  {
  // Add your code here to act upon Distance change
  Serial.println("--onDistanceChange");
}
void onTriggeredChange()
{
  Serial.print("--onTriggeredChange");
}
void onMaxThresholdChange()
{
  Serial.print("--onMaxThresholdChange");
}
void onMinThresholdChange()
{
  Serial.print("--onMinThresholdChange");
}
/*
  Since RandomTemperature is READ_WRITE variable, onRandomTemperatureChange()
is
  executed every time a new value is received from IoT Cloud.
*/
```


sketch_Task3.2C.ino arduino_secrets.h thingProperties.h

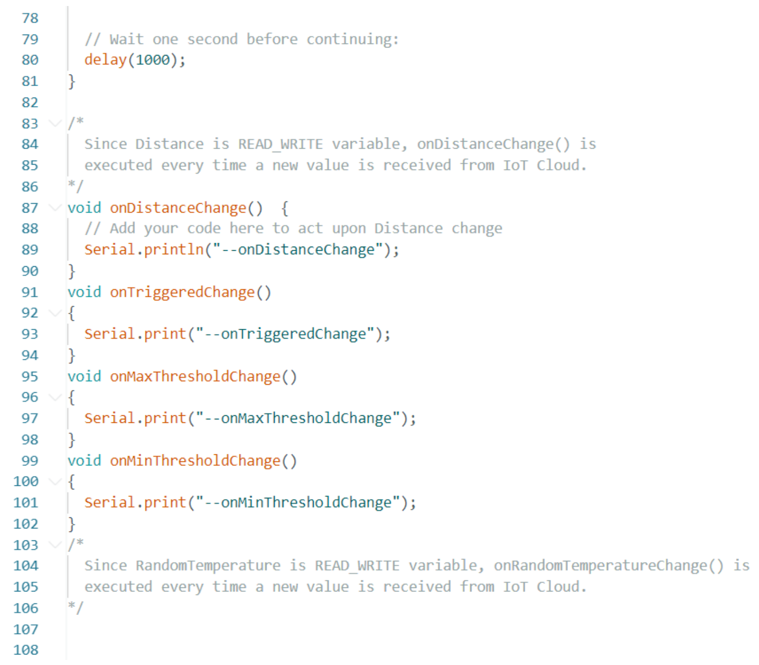
```
#include "thingProperties.h"

// Configure the PIN
const int trigger = 2;
const int echo = 3;


int getUltrasonicDistance(){
  // Function to retreive the distance reading of the ultrasonic sensor
  long duration;


  // Assure the trigger pin is LOW:
  digitalWrite(trigger, LOW);
  // Brief pause:
  delayMicroseconds(5);

  // Trigger the sensor by setting the trigger to HIGH:
  digitalWrite(trigger, HIGH);
  // Wait a moment before turning off the trigger:
  delayMicroseconds(10);
  // Turn off the trigger:
  digitalWrite(trigger, LOW);

```



```
  // Read the echo pin:
  duration = pulseIn(echo, HIGH);
  // Calculate the distance in centimeter (CM):
  distance = duration * 0.034 / 2;

  // Uncomment this line to return value in IN instead of CM:
  //distance = distance * 0.3937008

  // Return the distance read from the sensor:
  return distance;
}
void setup() {
  // Define inputs and outputs:
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
```



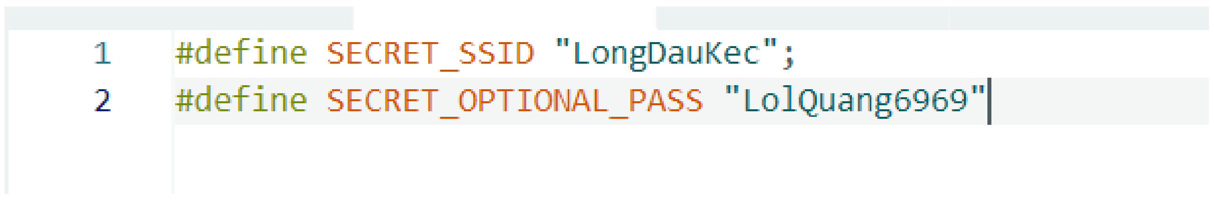
```
  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  // Your code here
  // Print the distance to the serial monitor:
  distance = getUltrasonicDistance();
  Serial.print("Distance: ");

  if (distance < min_threshold || distance > max_threshold) // If the object is too near the sensor  or exceeds the maximum threshold, it will trigger the alarm.
  {
    triggered = true;
  }
  else {
    triggered = false;
  }
  Serial.println(distance);

```



```
  
  // Wait one second before continuing:
  delay(1000);
}

/*
  Since Distance is READ_WRITE variable, onDistanceChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onDistanceChange()  {
  // Add your code here to act upon Distance change
  Serial.println("--onDistanceChange");
}
void onTriggeredChange()
{
  Serial.print("--onTriggeredChange");
}
void onMaxThresholdChange()
{
  Serial.print("--onMaxThresholdChange");
}
void onMinThresholdChange()
{
  Serial.print("--onMinThresholdChange");
}
/*
  Since RandomTemperature is READ_WRITE variable, onRandomTemperatureChange() is
  executed every time a new value is received from IoT Cloud.
*/
```

Arduino_secrets.h

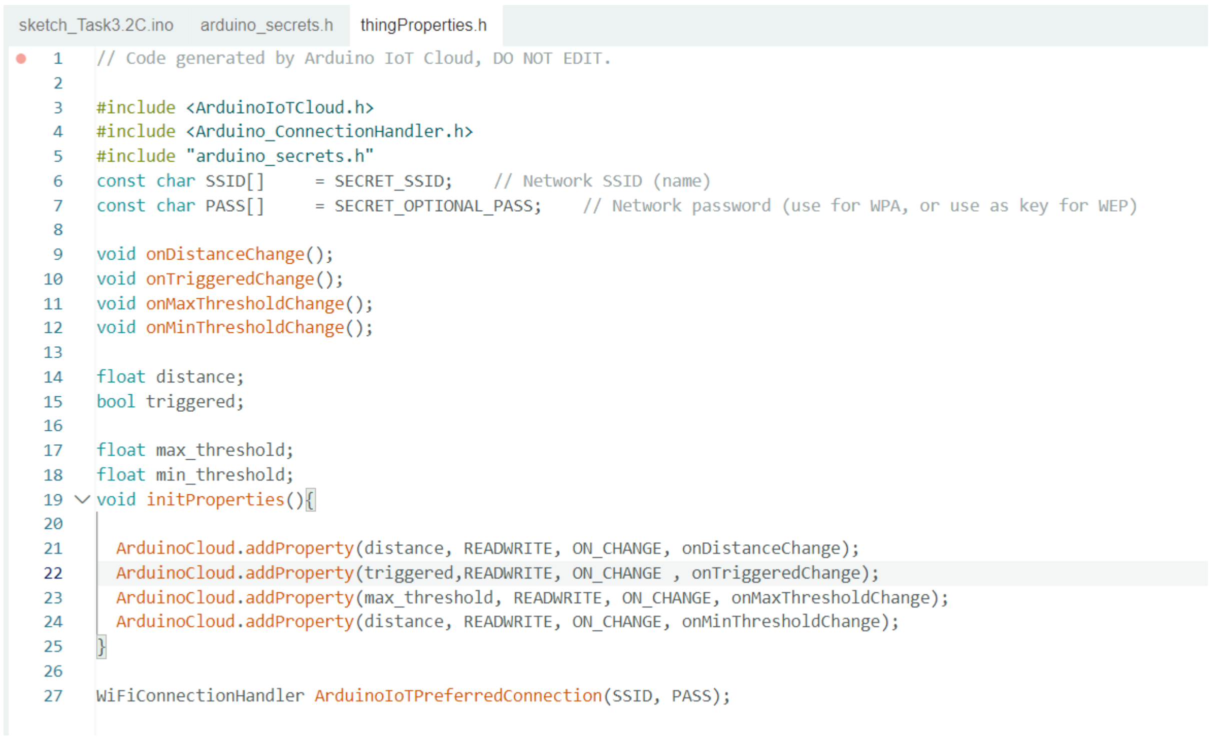
```
#define SECRET_SSID "LongDauKec";

#define SECRET_OPTIONAL_PASS  "LolQuang6969"
```



```
#define SECRET_SSID "LongDauKec";
#define SECRET_OPTIONAL_PASS "LolQuang6969"
```

thingProperties.h

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.


#include <ArduinoIoTCloud.h>

#include <Arduino_ConnectionHandler.h>

#include "arduino_secrets.h"

const char SSID[]     = SECRET_SSID;    // Network SSID (name)

const char PASS[]    = SECRET_OPTIONAL_PASS;    // Network password (use for WPA, or use as key for WEP)


void onDistanceChange();
void onTriggeredChange();
void onMaxThresholdChange();
void onMinThresholdChange();


float distance;
bool triggered;


float max_threshold;
float min_threshold;
void initProperties(){


  ArduinoCloud.addProperty(distance, READWRITE, ON_CHANGE, onDistanceChange);
  ArduinoCloud.addProperty(triggered,READWRITE, ON_CHANGE , onTriggeredChange);
  ArduinoCloud.addProperty(max_threshold, READWRITE, ON_CHANGE,
onMaxThresholdChange);
  ArduinoCloud.addProperty(distance, READWRITE, ON_CHANGE,
onMinThresholdChange);
}


WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```



sketch_Task3.2C.ino arduino_secrets.h thingProperties.h

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>
#include "arduino_secrets.h"
const char SSID[]     = SECRET_SSID;    // Network SSID (name)
const char PASS[]     = SECRET_OPTIONAL_PASS;    // Network password (use for WPA, or use as key for WEP)

void onDistanceChange();
void onTriggeredChange();
void onMaxThresholdChange();
void onMinThresholdChange();

float distance;
bool triggered;

float max_threshold;
float min_threshold;
void initProperties(){

  ArduinoCloud.addProperty(distance, READWRITE, ON_CHANGE, onDistanceChange);
  ArduinoCloud.addProperty(triggered,READWRITE, ON_CHANGE , onTriggeredChange);
  ArduinoCloud.addProperty(max_threshold, READWRITE, ON_CHANGE, onMaxThresholdChange);
  ArduinoCloud.addProperty(distance, READWRITE, ON_CHANGE, onMinThresholdChange);
}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

Basically the Arduino code will connect the variables distance, triggered, max_threshold and min_threshold to the corresponding widgets as mentioned earlier.

Video Demonstration Link:

https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=bebb0c6f-4d83-4718-8f71-b1cd004ddb17