CURIE Academy, Summer 2014
Lab 3: "Smart Door" IoT System

Sign-Off Sheet

NAME: _____

NAME: _____

NAME: _____

NAME: _____

NAME: _____

NAME: _____

DATE: _____

Sign-Off Milestone		TA Initials
Part 1.A	IoT Input Device: Button to LCD	
Part 1.B	IoT Input Device: Button to Cloud	
Part 2.A	IoT Output Device: Blinking LED	
Part 2.B	IoT Output Device: Cloud to LED	
Part 2	"Smart Door" IoT System	

CURIE Academy, Summer 2014
Lab Handout 3: "Smart Door" IoT System

Prof. Christopher Batten
School of Electrical and Computer Engineering
Cornell University

In this (short) lab assignment, you will build your first IoT system, and this system will serve as the foundation for your design project. For this lab assignment you will work in your design project group of 5–6 students. The lab includes three required parts. To complete the lab you should first divide into two subgroups with 2–3 students in each subgroup. One subgroup will focus on the first part, while the other subgroup focuses on the second part. Then the entire group will work on the final part together. In Part 1, you will build an IoT input device that monitors a button in the spirit of a door switch and uploads the door status to the cloud. In Part 2, you will build an IoT output device that periodically checks the cloud and displays the door status using an LED. In the final part you will put the IoT input and output devices together to create the full IoT system as illustrated in Figure 1.

Feel free to consult the lab notes as you work through the lab. For each part and various subparts you will need to have a teaching assistant observe the desired outcome and initial the appropriate line on the sign-off sheet. **Turn in your sign-off sheet at the end of the lab session.**

Before beginning, take a look at the materials on the lab bench which you will be using to complete the lab:

- Two Arduino-based IoT devices
- Push button for input IoT device
- LED and resistor for output IoT device
- Two screwdrivers for assembling/disassembling IoT devices
- Two wire cutter and strippers
- Two Arduino cheat sheets
- Two "Introduction to Arduino" booklets
- Workstation with black USB cable
- Flash drive for storing your code

You will also need to log into Xively and create three channels in your group's workspace. The channels should be named "door_status", "door_status_in", and "door_status_out". Consult the lab notes for more details on how to do this.

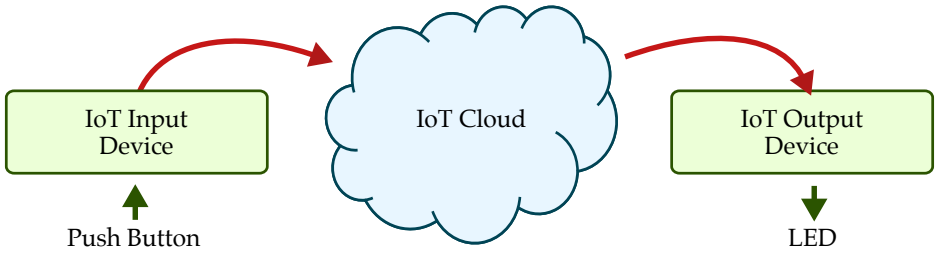

Figure 1: Diagram of "Smart Door" IoT System

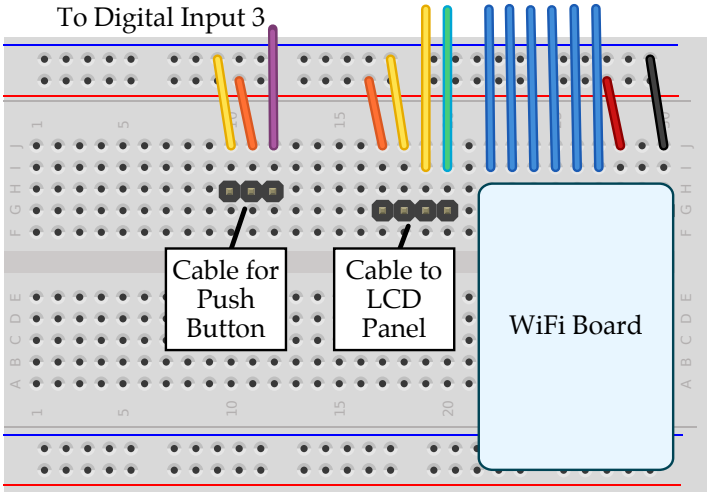

Figure 2: Breadboard Diagram for Wiring Push Button

1. IoT Input Device

You should divide into two subgroups. The first subgroup with 2–3 students should focus on this part of the lab where you will build an IoT input device that monitors a button in the spirit of a door switch and uploads the door status to the cloud. We will start by simply reading the button and displaying its status on the LCD panel, before moving onto uploading the door status to the cloud. Note that in this part, you will be using a different channel ("door_status_in") from what the other subgroup is using ("door_status_out"). This will enable you to test both parts of the system in isolation before combining them together.

1.A IoT Input Device: Button to LCD

As a first step, wire the push button to your IoT input device as shown in Figure 2. Then enter the code shown in Figure 3 to display the status of the push button on the LCD panel.

Sign-Off Milestone: Once you have verified that the door status is displayed correctly, demonstrate the system for an instructor.

1.B IoT Input Device: Button to Cloud

Now that we know the push button works, we will upload the door status to the cloud. Figure 4 illustrates how we can send the door status to the cloud using the `send_int` routine. **Note that you must use your group number instead of N when initializing** `curie_cloud`**!**. To verify that the door status is being upload to the cloud correctly, watch the "door_status_in" channel in Xively. Since we have a four second delay to avoid overloading the Xively servers, you will need to keep the button pressed for ten seconds or so to ensure that the system correctly uploads the new door status. If you would like to make your IoT input device more responsive, you could modify the code to wait until the button changes and then immediately upload the new value.

As mentioned in the lab notes, you must limit how fast you access the Xively servers to avoid being blocked. See the lab notes for more details.

Sign-Off Milestone: Once you have verified that the door status is being uploaded to the cloud correctly, demonstrate the system for an instructor.

```
// We need to include three libraries

#include <Curie.h>
#include <Wire.h>
#include <SPI.h>

// Global constants for pin assignments

int pin_button = 3;

// The setup routine runs once when you press reset

void setup()
{
  curie_lcd.begin();            // Initialize LCD panel
  pinMode( pin_button, INPUT ); // Configure pin_button as input
}

// The loop routine runs over and over again

void loop()
{
  // Read the push button's current value
  int door_status = digitalRead( pin_button );

  // Output door status to LCD panel
  curie_lcd.setCursor(0,0);
  curie_lcd.print("Door Status: ");
  curie_lcd.print(door_status);
}
```

Figure 3: Program to Update LCD Based on Push Button

```
// We need to include three libraries

#include <Curie.h>
#include <Wire.h>
#include <SPI.h>

// Global constants for pin assignments

int pin_button = 3;

// The setup routine runs once when you press reset

void setup()
{
  curie_cloud.begin( CURIE_CLOUD_GROUP(N) ); // Setup the cloud
  curie_lcd.clear();                         // Clear the LCD panel
  pinMode( pin_button, INPUT );              // Configure pin_button as input
}

// The loop routine runs over and over again

void loop()
{
  // Read the current door status
  int door_status = digitalRead( pin_button );

  // Output door status to LCD panel
  curie_lcd.setCursor(0,0);
  curie_lcd.print("Door Status: ");
  curie_lcd.print(door_status);

  // Send door status to the cloud
  curie_cloud.send_int( "door_status_in", door_status );

  // Wait 4 seconds to avoid overloading Xively servers
  delay(4000);
}
```

Figure 4: Program to Send Door Status To Cloud

2. IoT Output Device

You should divide into two subgroups. The second subgroup with 2–3 students should focus on this part of the lab where you will build an IoT output device that periodically checks the cloud and displays the door status using an LED. We will start by simply blinking the LED, before moving onto downloading the door status from the cloud. Note that in this part, you will be using a different channel ("door_status_out") from what the other subgroup is using ("door_status_in"). This will enable you to test both parts of the system in isolation before combining them together.

2.A IoT Output Device: Blink LED

As a first step, wire the LED as shown in Figure 5. Then enter the code shown in Figure 6 to blink both the LED and the LCD panel.

Sign-Off Milestone: Once you have verified that the LED and LCD panel are blinking correctly, demonstrate the system for an instructor.

2.B IoT Output Device: Cloud to LED

Now that we know the LED works, we will receive the door status from the cloud. Figure 7 illustrates how we can receive the door status from the cloud using the `recv_int` routine. **Note that you must use your group number instead of N when initializing** `curie_cloud`**!**. To verify that the door status is being received from the cloud correctly, you can directly modify the current value of the "door_status_out" channel in Xively. After modifying the current value, your IoT output device should reflect the change within a few seconds.

As mentioned in the lab notes, you must limit how fast you access the Xively servers to avoid being blocked. See the lab notes for more details.

Sign-Off Milestone: Once you have verified that the LED and LCD panel are correctly displaying the door status received from the cloud, demonstrate the system for an instructor.

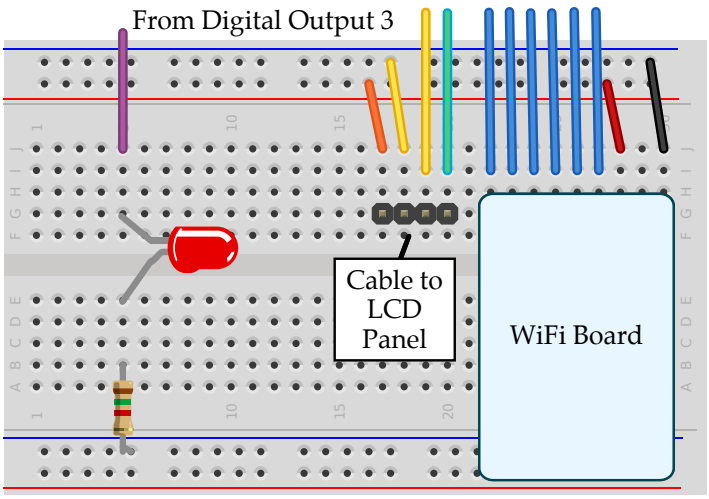

Figure 5: Breadboard Diagram for Wiring LED

```
// We need to include three libraries

#include <Curie.h>
#include <Wire.h>
#include <SPI.h>

// Global constants for pin assignments

int pin_led = 3;

// The setup routine runs once when you press reset

void setup()
{
  curie_lcd.begin();           // Initialize LCD panel
  pinMode( pin_led, OUTPUT  ); // Configure pin_led as output
}

// The loop routine runs over and over again

void loop()
{
  digitalWrite( pin_led, HIGH );

  curie_lcd.setCursor(0,0);
  curie_lcd.print("LED is on ");

  delay(1000);

  digitalWrite( pin_led, LOW );

  curie_lcd.setCursor(0,0);
  curie_lcd.print("LED is off");

  delay(1000);
}
```

Figure 6: Program to Blink LED and LCD

```
// We need to include three libraries

#include <Curie.h>
#include <Wire.h>
#include <SPI.h>

// Global constants for pin assignments

int pin_led = 3;

// The setup routine runs once when you press reset

void setup()
{
  curie_cloud.begin( CURIE_CLOUD_GROUP(N) ); // Setup the cloud
  curie_lcd.clear();                         // Clear the LCD panel
  pinMode( pin_led, OUTPUT );                // Configure pin_button as input
}

// The loop routine runs over and over again

void loop()
{
  // Display "checking cloud" on LCD panel
  curie_lcd.setCursor(0,1);
  curie_lcd.print("checking cloud");

  // Receive door status from the cloud
  int door_status = curie_cloud.recv_int( "door_status_out" );

  // Clear LCD panel
  curie_lcd.setCursor(0,1);
  curie_lcd.print("              ");

  // Update LED and LCD panel based on door status
  if ( door_status == 1 ) {
    digitalWrite( pin_led, HIGH );
    curie_lcd.setCursor(0,0);
    curie_lcd.print("Door is closed");
  }
  else {
    digitalWrite( pin_led, LOW );
    curie_lcd.setCursor(0,0);
    curie_lcd.print("Door is open  ");
  }

  // Wait 4 seconds to avoid overloading Xively servers
  delay(4000);
}
```

Figure 7: Program to Receive Door Status From Cloud

3. IoT System

In this final part, you will now link the IoT input device to the IoT output device so the door status as indicated by the push button is reflected by the LED. Assuming you have tested the input and output devices in isolation this part should be relatively straight forward. All you need to do is modify the code running on both devices to access the same "door_status" channel. Try running the two devices off of the battery and move to opposite sides of the lab to verify that the system is working.

Sign-Off Milestone: Once you have verified that your system works, demonstrate the system for an instructor.