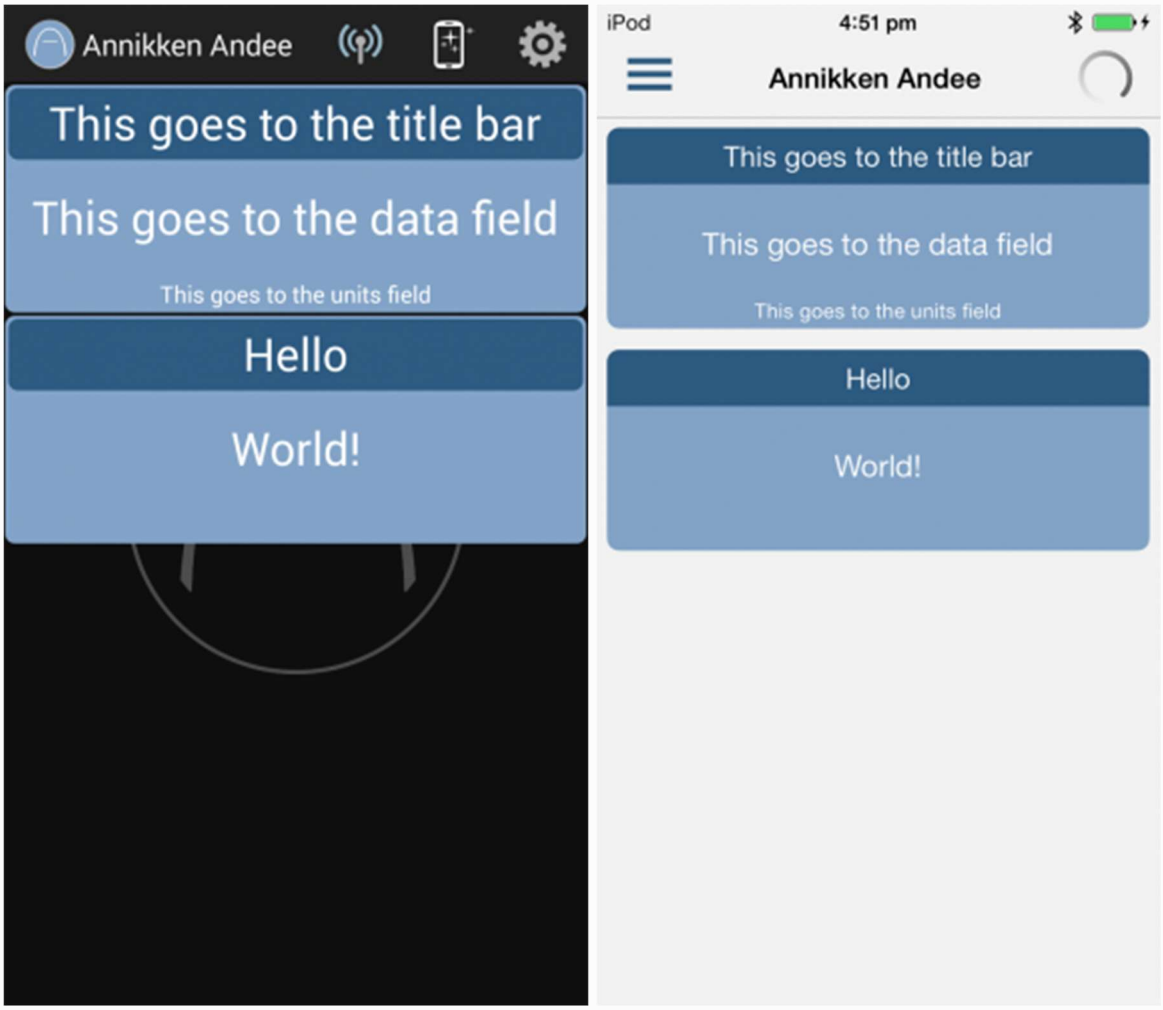
Dans ce document, vous trouverez un ensemble de petits TPs vous permettant d'utiliser la carte Annikken Andee avec les systèmes d'exploitation mobiles iOS et Android

Table des matières

Leçon n°1 :

Créer vos 1ères boites d'affichage.

Pour cette première leçon, nous allons voir comment créer des boites d'affichages en utilisant la carte Annikken Andee.
Voici ci-dessous une représentation de l'interface utilisateur que nous allons créer. Le code vous est fourni plus loin.



Top of the Code

Les bibliothèques ci-dessous doivent toujours être inclues ; la carte Annikken Andee en a besoin pour travailler avec Arduino.

```
#include <SPI.h>

#include <Andee.h>



// Every object that appears on your smartphone's screen

// needs to be declared like this:

AndeeHelper objectA;

AndeeHelper objectB;

// We're creating two objects here
```

setup()

La fonction setup() signifie à Arduino quoi faire quand le programme démarre.

```
void setup()

{

  Andee.begin();  // Setup communication between Annikken Andee and Arduino

  Andee.clear();  // Clear the screen of any previous displays

  setInitialData(); // Define object types and their appearance

}
```

setInitialData()

Cette fonction permet de définir les styles et l'apparence de tous les objets de votre smartphone.

```
void setInitialData()
{
  //// Let's draw the first object! ////////////////////////////////////////
  objectA.setId(0);  // Each object must have a unique ID number
  objectA.setType(DATA_OUT);  // This defines your object as a display box
  objectA.setLocation(0, 0, FULL); // Sets the location and size of your object
  /* setLocation (row, col, size)
     Row: From 0 (top-most) to 3
     Col: From 0 (left-most) to 9. If there are too many objects on that row, you can
          scroll from left to right.
     Size: The following sizes are available for you to chooose:
     FULL, HALF, ONE_THIRD, ONE_QUART, TWO_THIRD, THREE_QUART */
  objectA.setTitle("This goes to the title bar");
  objectA.setData("This goes to the data field");
  objectA.setUnit("This goes to the units field"); // Optional
  //// Let's draw the second object! ///////////////////////////////////////
  objectB.setId(1); // Don't forget to give it a unique ID number
  objectB.setType(DATA_OUT); // Another display box
  objectB.setLocation(1,0,FULL); // Second row, left-most, full size
  objectB.setTitle("Hello");
  objectB.setData("World!");
}
```

loop()

Arduino répètera en boucle les instructions jusqu'à ce que vous lui demandiez d'arrêter.

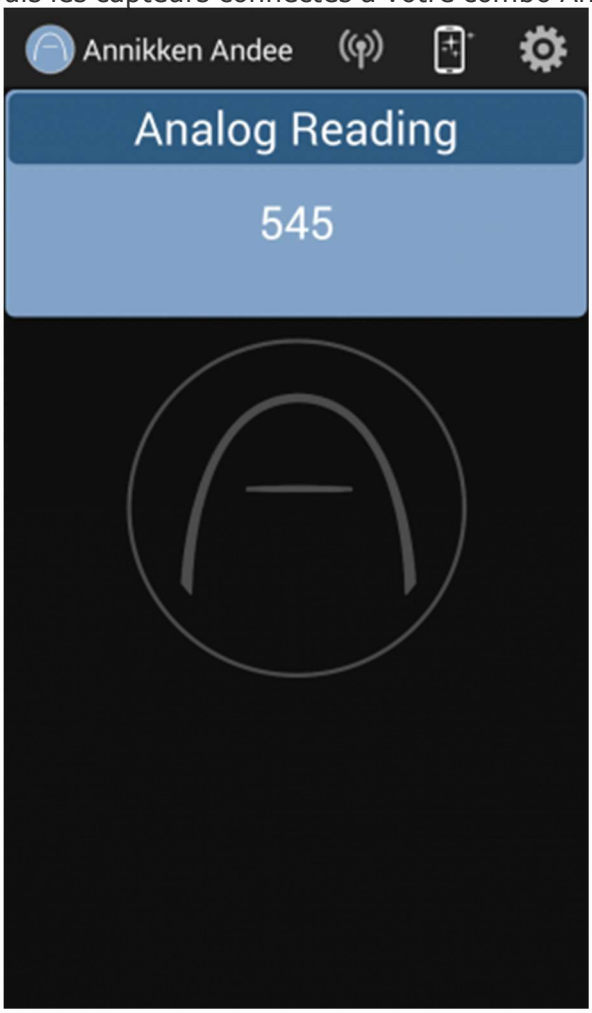
```
void loop()

{

  objectA.update(); // Call update() to refresh the display on your screen

  objectB.update(); // If you forgot to call update(), your object won't appear



  // A short delay is necessary to give Andee time to communicate with the smartphone

  delay(500);

}
```

Leçon n°2:

Afficher une information en entrée analogique.

Vous pouvez retrouver cette leçon dans l'IDE Arduino (File -> Examples -> Andee). Si ce n'est pas le cas, vous devez au préalable installer Andee Library for Arduino IDE.

Dans cette leçon, nous allons vous montrer comment afficher facilement les données d'un capteur analogique depuis les capteurs connectés à votre combo Annikken Andee/Arduino.



Top of the Code

Ce code s'utilise pour tous les capteurs.

Les bibliothèques ci-dessous doivent toujours être inclues ; la carte Annikken Andee en a besoin pour travailler avec Arduino.

```
#include <SPI.h>

#include <Andee.h>

// We'll creating one object to display the analog input signal

AndeeHelper analogDisplay;

// We'll use Analog Input Pin A0 to read our analog input.

// Change the pin number if you are using another pin.

const int analogInputPin = A0;
```

setup()

La fonction setup() signifie à Arduino quoi faire quand le programme démarre.

```
void setup()

{

  Andee.begin();  // Setup communication between Annikken Andee and Arduino

  Andee.clear();  // Clear the screen of any previous displays

  setInitialData(); // Define object types and their appearance

}
```

setInitialData()

Cette fonction permet de définir les styles et l'apparence de tous les objets de votre smartphone.

```
void setInitialData()
{
  // Only one display box this time
  analogDisplay.setId(0);  // Each object must have a unique ID number
  analogDisplay.setType(DATA_OUT);  // This defines your object as a display box
  analogDisplay.setLocation(0, 0, FULL); // Sets the location and size of your object
  analogDisplay.setTitle("Analog Reading");
  analogDisplay.setData(""); // We'll update it with new analog data later.
}
```

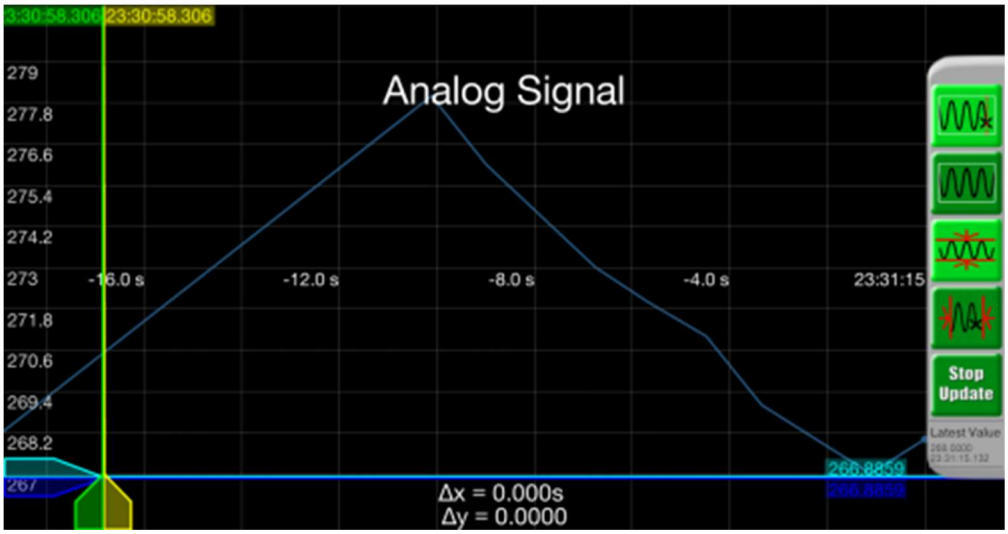
loop()

Arduino répètera en boucle les instructions jusqu'à ce que vous lui demandiez d'arrêter.

```
void loop()
{
  // Read value from analog pin and store it in an int variable
  int reading = analogRead(analogInputPin);
  analogDisplay.setData(reading); // Set the display box with new data value
  analogDisplay.update(); // Update the display to show the new value
  // A short delay is necessary to give Andee time to communicate with the smartphone
  delay(500);
}
```

Visualisation graphique

Savez-vous que, lorsque vous restez appuyer sur la boite d'affichage, votre smartphone dessinera un graphe à partir des données émises par le ou les capteurs ?



Leçon n°3:

Varier les couleurs en fonction des données du capteur.

Dans cette leçon, nous allons voir comment modifier facilement et automatiquement la couleur de vos boites d'affichage en fonction des changements de données d'un capteur. Ceci est très utile pour interpréter les données reçues d'un capteur.

Top of the Code

Les bibliothèques ci-dessous doivent toujours être inclues ; la carte Annikken Andee en a besoin pour travailler avec Arduino.

```
#include <SPI.h>

#include <Andee.h>



// We'll create a display to show you how you can

// change the colours of your display boxes according to the

// sensor readings

AndeeHelper colourDisplay;



// We'll use Analog Input Pin A0 to read in the analog input

const int analogInputPin = A0;
```

setup()

La fonction setup() signifie à Arduino quoi faire quand le programme démarre.

```
void setup()
{
  Andee.begin();  // Setup communication between Annikken Andee and Arduino
  Andee.clear();  // Clear the screen of any previous displays
  setInitialData(); // Define object types and their appearance
}
```

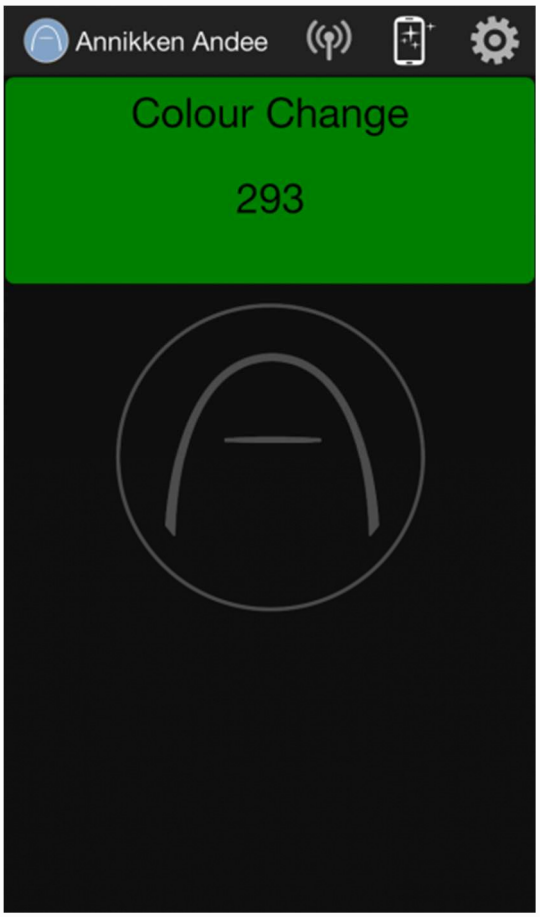
setInitialData()

Cette fonction permet de définir les styles et l'apparence de tous les objets de votre smartphone.

```
void setInitialData()
{
  // Let's draw colourDisplay first!
  colourDisplay.setId(0);  // Each object must have a unique ID number
  colourDisplay.setType(DATA_OUT);  // This defines your object as a display box
  colourDisplay.setLocation(0, 0, FULL); // Sets the location and size of your object
  colourDisplay.setTitle("Colour Change"); // Sets title
  colourDisplay.setData(""); // We'll update it with new analog data later.
}
```

loop()

Le code ci-dessous permet de changer la couleur (et la couleur du titre) de la boite d'affichage en vert, jaune ou rouge, en function des données du capteur.



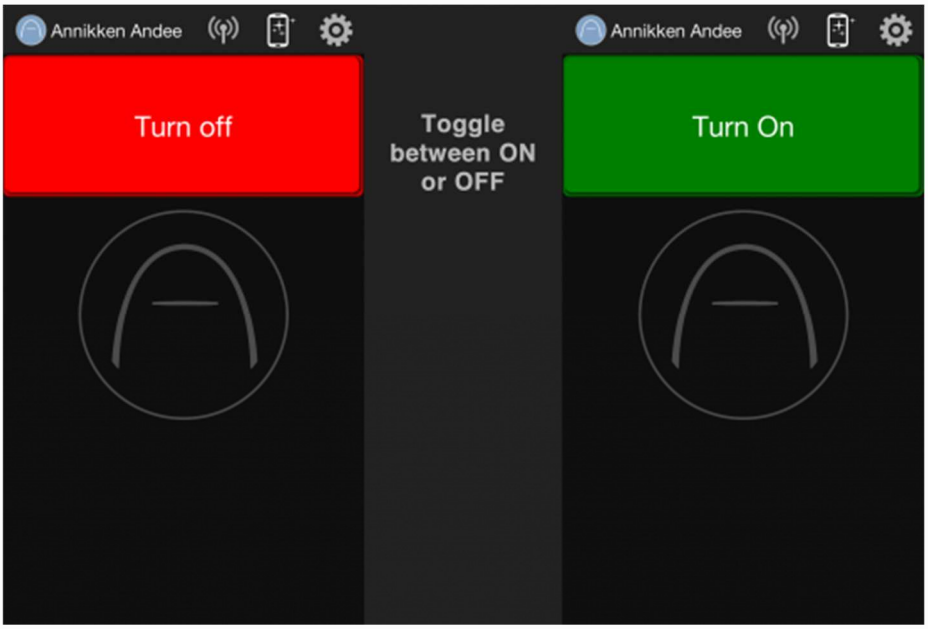

```
void loop()
{
  // Read values from analog pins and store it in int variables
  int reading = analogRead(analogInputPin);

  if(reading < 300) // Feel free to change the value
  {
    colourDisplay.setColor(GREEN); // Sets background colour
```

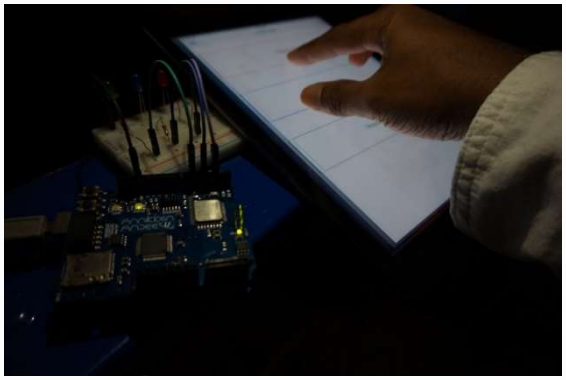
```
  Or you can use the ARGB colour code to set your own colour, in the form:
  discreteDisplay.setColor("FF00FF00"); */

  colourDisplay.setTextColor(BLACK); // Sets font colour.

  colourDisplay.setTitleColor(GREEN); // Sets title background colour
  colourDisplay.setTitleTextColor(BLACK); // Sets title font colour
 }
 else if(reading < 350) // Feel free to change the value
 {
  colourDisplay.setColor(YELLOW);
  colourDisplay.setTextColor(BLACK);
  colourDisplay.setTitleColor(YELLOW);
  colourDisplay.setTitleTextColor(BLACK);
 }
 else
 {
  colourDisplay.setColor(RED);
  colourDisplay.setTextColor(WHITE);
  colourDisplay.setTitleColor(RED);
  colourDisplay.setTitleTextColor(WHITE);
 }

 colourDisplay.setData(reading); // Set the display box with new data value
 colourDisplay.update(); // Update the display to show the new value

 // A short delay is necessary to give Andee time to communicate with the smartphone
 delay(500);
}
```

Leçon n°4:

Créer un bouton commutateur.

Un bouton commutateur permet de passer d'un état à l'autre, par exemple de commuter entre 'Allumé' et 'Eteint'.





Top of the Code

Les bibliothèques ci-dessous doivent toujours être inclues ; la carte Annikken Andee en a besoin pour travailler avec Arduino.

```
#include <SPI.h>

#include <Andee.h>

// We'll just create one button to toggle

AndeeHelper togglebutton;

int state; // This variable will store the current state
```

setup()

La fonction setup() signifie à Arduino quoi faire quand le programme démarre.

```
void setup()

{

  Andee.begin();  // Setup communication between Annikken Andee and Arduino

  Andee.clear();  // Clear the screen of any previous displays

  setInitialData(); // Define object types and their appearance



  state = 0; // Initialise your state to zero

}
```

setInitialData()

Cette fonction permet de définir les styles et l'apparence de tous les objets de votre smartphone.

```
void setInitialData()
{
  // Let's draw a toggle button
  togglebutton.setId(0); // Don't forget to assign a unique ID number
  togglebutton.setType(BUTTON_IN); // Defines object as a button
  togglebutton.setLocation(0,0,FULL);
  togglebutton.setTitle("Turn On"); // Sets the initial words for button
  togglebutton.setColor(GREEN);
  // You can't use setData() and setUnit() for buttons.
}
```

loop()

Arduino répètera en boucle les instructions jusqu'à ce que vous lui demandiez d'arrêter

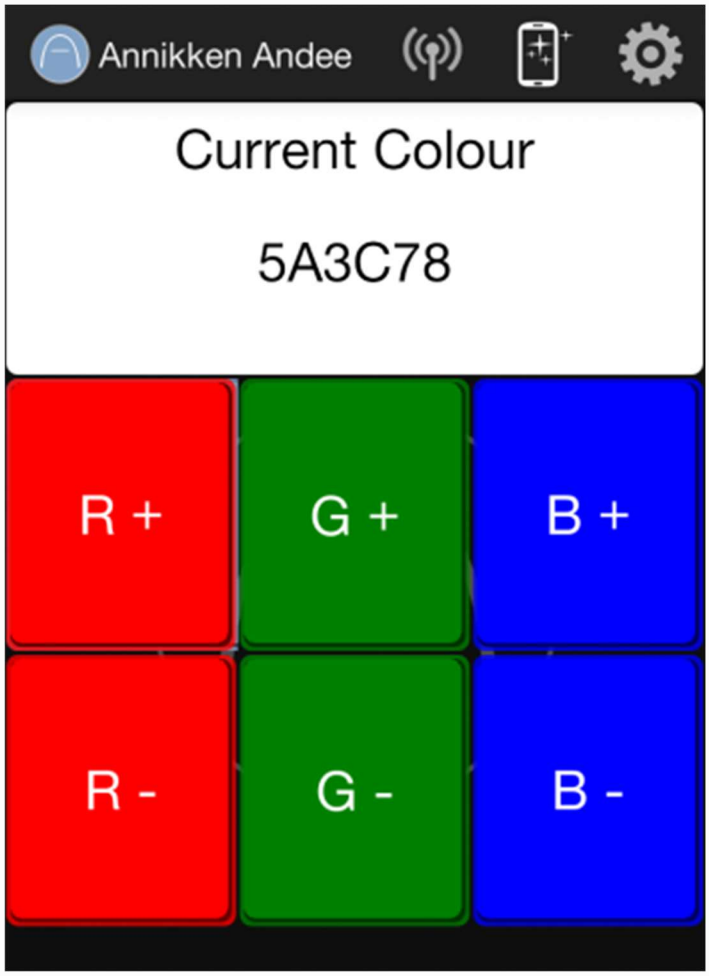
```
void loop()
{
  // Here's how you code the button action
  if( togglebutton.isPressed() )
  {
    // Prevent user from accidentally pressing the button again
```

```
    // until Arduino has sent an acknowledgement
    togglebutton.ack();

    if(state == 0)
    {
      togglebutton.setTitle("Turn off");
      state = 1; // Change state
      togglebutton.setColor(RED);
      // Add additional actions here to turn on something
    }
    else
    {
      togglebutton.setTitle("Turn on");
      state = 0; // Change state
      togglebutton.setColor(GREEN);
      // Add additional actions here to turn off something
    }
  }

  togglebutton.update(); // Update your button to reflect the change in state

  delay(500); // Always leave a short delay for Bluetooth communication
}
```

Leçon n°5 :

Utiliser des boutons pour contrôler des LEDs de couleur.

La LED RGB, également appelée LED tricolore, est composées de 3 anodes rouge, verte et bleue. Vous pouvez ajuster l'intensité de chaque couleur afin d'obtenir une palette de couleurs variées.
Dans cette leçon, nous allons voir comment utiliser votre smartphone pour contrôler la couleur de la LED RGB en appuyant sur des boutons.

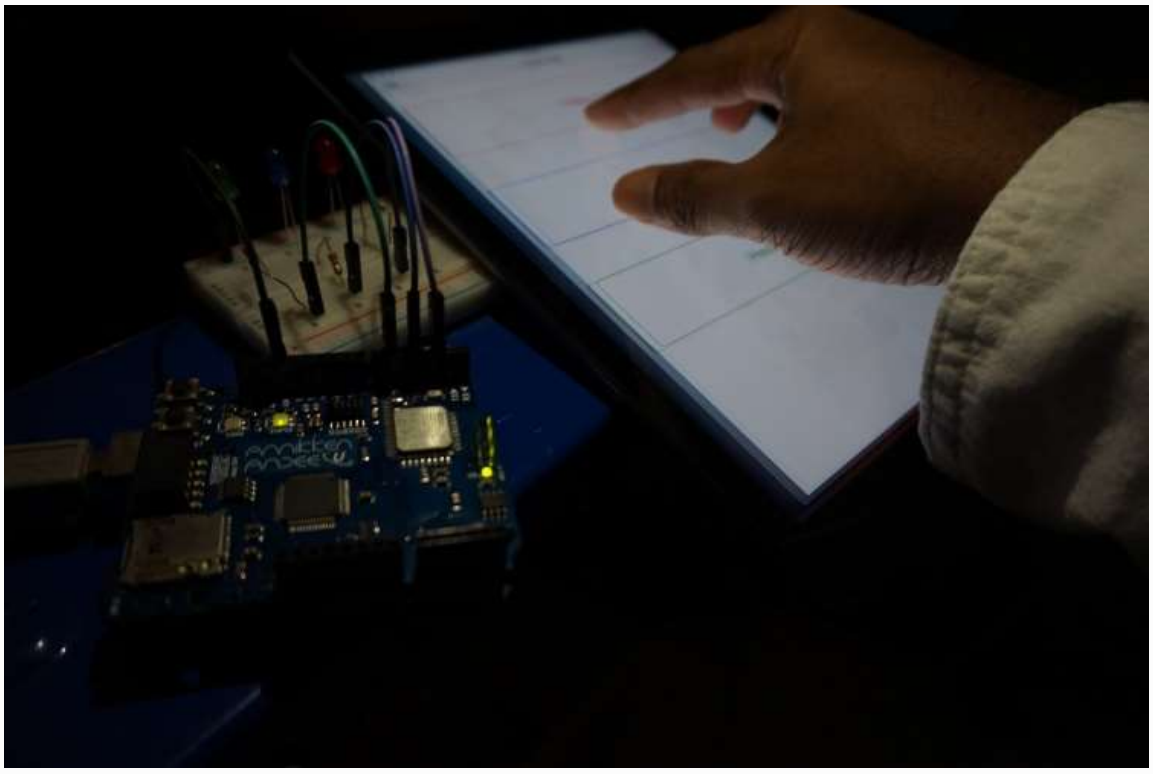


Note: *Le code que nous avons programmé ici ne s'applique qu'aux LEDs RGB standards. Dans une LED RGB standard, les trois couleurs partagent la même broche GND. Le câblage et le codage diffèreront légèrement si vous utilisez une LED anode standard (c'est-à-dire que les trois couleurs partagent la même alimentation, mais ont chacune leur propre broche).*

== Attention, pour ce TP, vous devez avoir au minimum les versions :

Annikken Andee for iOS - V2.3 (16 Mar 2016)

Annikken Andee U - V1.4 (16 Mar 2016) ==



Voici ci-dessous le programme au complet.

Top of the Code

```
#include <SPI.h>
#include <Andee.h>

// We'll need 6 buttons to increase/decrease the red, green
// and blue channels of our RGB LED
AndeeHelper displayCurrentColour;
AndeeHelper buttonRup; // Buttons to adjust Red
AndeeHelper buttonRdown;
AndeeHelper buttonGup; // Buttons to adjust Green
AndeeHelper buttonGdown;
AndeeHelper buttonBup; // Buttons to adjust Blue
AndeeHelper buttonBdown;

// To get various colours from the RGB LED, we'll need to connect
// the LED to pins capable of analog output. On the Arduino Uno,
// pins capable of analog output are Pins 3, 5, 6, 9, 10.
// Do not use Pins 8, 11, 12, 13 (Arduino Uno layout) as Andee
// is using them. Strange things will happen if you do.
const int pinR = 6; // Red is connected to pin 6
const int pinG = 5; // Green is connected to pin 5
const int pinB = 3; // Blue is connected to pin 3

// We'll need to store the intensity levels of each channel
// Note: Analog output is only capable of producing a range from 0 to 255
int r = 0; // Red channel
int g = 0; // Green channel
int b = 0; // Blue channel

int resolution = 30; // Set the amount to increase/decrease for each button press
char colour[7] = "000000"; // The initial colour of our display
```

setup()

```
void setup()
{
  Andee.begin();  // Setup communication between Annikken Andee and Arduino
  Andee.clear();  // Clear the screen of any previous displays
  setInitialData(); // Define object types and their appearance

  pinMode(pinR, OUTPUT); // Set pins to output mode
  pinMode(pinG, OUTPUT);
  pinMode(pinB, OUTPUT);
}
```

setInitialData()

Pas de panique ! Le code ci-dessous va vous paraître long au premier regard ; cependant, celui-ci se répète plusieurs foir pour définir les différents écrans des boites d'affichages.

```
void setInitialData()
{
  displayCurrentColour.setId(0); // Our display box
  displayCurrentColour.setType(DATA_OUT);
  displayCurrentColour.setLocation(0,0,FULL);
  displayCurrentColour.setTitle("Current Colour");
  displayCurrentColour.setTitleColor(WHITE); // Set title to white
  displayCurrentColour.setTitleTextColor(TEXT_DARK); // Set title font to black
  displayCurrentColour.setColor(WHITE);
  displayCurrentColour.setTextColor(TEXT_DARK);
  displayCurrentColour.setData(colour); // Show the RGB colour code.

  buttonRup.setId(1); // Button to increase red level
  buttonRup.setType(BUTTON_IN);
  buttonRup.setLocation(1,0,ONE_THIRD);
  buttonRup.setTitle("R +");
  buttonRup.setColor(RED); // Set button to red
  buttonRup.requireAck(false); // You need this line to allow for multiple button presses
```

```
  buttonRdown.setId(2); // Button to decrease red level
  buttonRdown.setType(BUTTON_IN);
  buttonRdown.setLocation(2,0,ONE_THIRD);
  buttonRdown.setTitle("R -");
  buttonRdown.setColor(RED); // Set button to red
  buttonRdown.requireAck(false); // You need this line to allow for multiple button
presses


  buttonGup.setId(3); // Button to increase green level
  buttonGup.setType(BUTTON_IN);
  buttonGup.setLocation(1,1,ONE_THIRD);
  buttonGup.setTitle("G +");
  buttonGup.setColor(GREEN); // Set button to green
  buttonGup.requireAck(false); // You need this line to allow for multiple button presses


  buttonGdown.setId(4); // Button to decrease green level
  buttonGdown.setType(BUTTON_IN);
  buttonGdown.setLocation(2,1,ONE_THIRD);
  buttonGdown.setTitle("G -");
  buttonGdown.setColor(GREEN); // Set button to green
  buttonGdown.requireAck(false); // You need this line to allow for multiple button
presses


  buttonBup.setId(5); // Button to increase blue level
  buttonBup.setType(BUTTON_IN);
  buttonBup.setLocation(1,2,ONE_THIRD);
  buttonBup.setTitle("B +");
  buttonBup.setColor(BLUE); // Set button to blue
  buttonBup.requireAck(false); // You need this line to allow for multiple button presses


  buttonBdown.setId(6); // Button to decrease blue level
  buttonBdown.setType(BUTTON_IN);
  buttonBdown.setLocation(2,2,ONE_THIRD);
  buttonBdown.setTitle("B -");
  buttonBdown.setColor(BLUE);  // Set button to blue
  buttonBdown.requireAck(false); // You need this line to allow for multiple button
presses
}
```

loop()

```
void loop()
{
  // This is the most important chunk of code. Each analogWrite() statement sets the intensity
  // of each colour channel.
  analogWrite(pinR, r);
  analogWrite(pinG, g);
  analogWrite(pinB, b);


  // Buttons used to adjust the colour channel intensity. We will employ
  // the press-and-hold button method to control the RGB LEDs
  if( buttonRup.getButtonPressCount() > 0 ) // Red button up will increase red intensity
  {
    r = r+resolution; // Increase the intensity by a fixed amount (specified above)
    if(r>255) r = 255; // 255 is the max value. If it goes beyond that, set the value to 255.
  }
  if( buttonRdown.getButtonPressCount() > 0 ) // Red button down will decrease red intensity
  {
    r = r-resolution; // Decrease the intensity by a fixed amount (specified above)
    if(r<1) r = 0; // 0 is the min value. If it goes under that, set the value to 0.
  }
  if( buttonGup.getButtonPressCount() > 0 ) // Green button up will increase green intensity
  {
    g = g+resolution;
    if(g>255) g = 255;
  }
  if( buttonGdown.getButtonPressCount() > 0 ) // Green button down will decrease green intensity
  {
    g = g-resolution;
    if(g<1) g = 0;
  }
```

```
  if( buttonBup.getButtonPressCount() > 0 ) // Blue button up will increase blue intensity
  {
    b = b+resolution;
    if(b>255) b = 255;
  }
  if( buttonBdown.getButtonPressCount() > 0 ) // Blue button down will decrease blue
intensity
  {
    b = b-resolution;
    if(b<1) b = 0;
  }


  sprintf(colour, "%02X%02X%02X", r,g,b); // Convert to an RGB colour code string
  displayCurrentColour.setData(colour); // Show user the RGB colour code on the
smartphone


  displayCurrentColour.update(); // Update screen
  buttonRup.update();
  buttonRdown.update();
  buttonGup.update();
  buttonGdown.update();
  buttonBup.update();
  buttonBdown.update();



  delay(500); // Always leave a short delay for Bluetooth communication
}
```

Leçon n°6 :

Modifier le nom du composant Bluetooth Andee.

Si vous voulez modifier le nom du composant Bluetooth Andee, voici comment procéder:

Top of the Code

```
#include <Andee.h> // Don't forget the necessary libraries
#include <SPI.h>

// This is where you change your device name:
char newBluetoothName[] = "Hello New Device Name"; // New device name
char cmdReply[64]; // String buffer
char commandString[100]; // String to store the new device name and device command into one
```

setup()

```
void setup()
{
  Andee.begin();
  Andee.clear();

  // We need to combine the new device name with the device command
  sprintf(commandString, "SET BT NAME %s", newBluetoothName);
  // Send command to change device name
  Andee.sendCommand(commandString, cmdReply);
}
```

loop()

```
void loop()

{

  // Disconnect user - there's nothing to do here anyway

  if(Andee.isConnected())

    Andee.disconnect();

}
```

Leçon n°7:

Envoyer un SMS

Savez-vous que vous pouvez utiliser Annikken Andee pour envoyer un SMS? Voici comment procéder :

Compatibilité

Cette leçon n'est réalisable que sous Android du fait des restrictions sécuritaires imposées par Apple.

A noter

Vous pourrez programmer votre combo Arduino/Annikken Andee pour envoyer automatiquement un SMS en fonction des données lues par un capteur. Dans ce cas précis, il est nécessaire d'utiliser une option qui limite l'envoi répétitif des données par le combo Arduino/Annikken Andee.

Top of the Code

```
#include <SPI.h>
#include <Andee.h>

// We'll use a button to send the message
AndeeHelper sendMessage;
AndeeHelper SMSobject; // The message and recipient must be declared as an object

// You can use the text input button to get the user to set the
// recipient number and even the message itself!
char messageRecipient[] = "+6587654321";
char message[] = "Hello World!";
```

setup()

```
void setup()
{
  Andee.begin();  // Setup communication between Annikken Andee and Arduino
  Andee.clear();  // Clear the screen of any previous displays
  setInitialData(); // Define object types and their appearance
}
```

setInitialData()

```
void setInitialData()
{
  sendMessage.setId(0);
  sendMessage.setType(BUTTON_IN);
  sendMessage.setLocation(0,0,FULL);
  sendMessage.setTitle("Send SMS");

  SMSobject.setId(1);
  SMSobject.setType(SMS_SENDER); // Sets object as an SMS object
  SMSobject.setRecipient(messageRecipient);
  SMSobject.setMessage(message);
}
```

loop()

```
void loop()
{
  if( sendMessage.isPressed() ) // When user presses the send button on phone
  {
    sendMessage.ack(); // Acknowledge button press
    SMSobject.send(); // Sends the SMS to the recipient
  }

  sendMessage.update();
  // Do not update SMS objects!

  delay(500); // Always leave a short delay for Bluetooth communication
}
```

Leçon n°8:

Faire parler votre smartphone sous Android.

Savez-vous que vous pouvez utiliser la fonction 'Texte-parole' (TTS) d'Android pour demander à Annikken Andee de lire un texte ? D'ailleurs, vous pouvez aller plus loin que la simple lecture d'un texte ! Cette fonction peut servir à lire les données reçues par un capteur sans que vous n'ayez à jeter un œil sur votre téléphone. C'est ce que nous allons voir ici!

Vérifiez que votre smartphone n'est pas en mode silencieux avant de commercer cette leçon et contrôler le volume sonore de votre appareil.
Si cela ne fonctionne pas, vous devez configure manuellement l'option 'Texte-Parole'. Pour cela, allez sur l'application Andee et ouvrez 'Settings'. Appuyez sur le bouton 'Setup' dans la section TTS.

Top of the Code

```
#include <SPI.h>
#include <Andee.h>

AndeeHelper displaybox;
AndeeHelper button;
AndeeHelper speechObject; // You need to create a speech object for the phone to talk
```

setup()

```
void setup()
{
  Andee.begin();  // Setup communication between Annikken Andee and Arduino
  Andee.clear();  // Clear the screen of any previous displays
  setInitialData(); // Define object types and their appearance
}
```

setInitialData()

```
void setInitialData()
{
  displaybox.setId(0);
  displaybox.setType(DATA_OUT);
  displaybox.setTitle("Text to Speech");
  displaybox.setData("Be sure to unmute your phone to hear your phone talk!");

  button.setId(1); // Don't forget to assign a unique ID number
  button.setType(BUTTON_IN); // Defines object as a button
  button.setLocation(1,0,FULL);
  button.setTitle("Say something!");
  button.setColor(THEME_RED_DARK);

  speechObject.setId(2);
  speechObject.setType(TTS); // Defines object as a Text-to-Speech object
}
```

loop()

```
void loop()
{
  if( button.isPressed() )
  {
    button.ack();
    // Use updateData() to get the phone to talk!
    speechObject.updateData("Confucius say: Man run in front of car get tired,");
    speechObject.updateData("man run behind car get exhausted.");
    // You will need to break your sentence into multiple lines if your lines are too long.
    // Each speech object can only handle up to 40 characters of text at a time.
    // Punctuation will also affect the way your phone vocalises the text.
  }

  displaybox.update();
  button.update();
  // Do not update the speechObject!

  delay(500); // Always leave a short delay for Bluetooth communication
}
```

Leçon n°9:

Faire parler votre smartphone sous iOS.

La function 'Texte-Parole' (TTS) d'Apple ressemble beaucoup à celle d'Android. Vous pouvez créer votre composant iOS pour faire parler votre smartphone via Annikken Andee. Vous pouvez également ajuster l'accent ou encore le rythme de la voix !

Vérifiez que votre smartphone n'est pas en mode silencieux avant de commercer cette leçon et contrôler le volume sonore de votre appareil.

Top of the Code

```
#include <SPI.h>
#include <Andee.h>

AndeeHelper displaybox;
AndeeHelper button;
AndeeHelper speechObject; // You need to create a speech object for the phone to talk
```

setup()

```
void setup()
{
  Andee.begin();  // Setup communication between Annikken Andee and Arduino
  Andee.clear();  // Clear the screen of any previous displays
  setInitialData(); // Define object types and their appearance
}
```

setInitialData()

Pour les accents, vous avez le choix entre US, GREAT_BRITON, AUSTRALIA, IRELAND et SOUTH_AFRICA.

```
void setInitialData()
{
  displaybox.setId(0);
  displaybox.setType(DATA_OUT);
  displaybox.setTitle("Text to Speech");
  displaybox.setData("Be sure to unmute your phone to hear your phone talk!");

  button.setId(1); // Don't forget to assign a unique ID number
  button.setType(BUTTON_IN); // Defines object as a button
  button.setLocation(1,0,FULL);
  button.setTitle("Say something!");
  button.setColor(THEME_RED_DARK);

  speechObject.setId(2);
  speechObject.setType(TTS); // Defines object as a Text-to-Speech object
  speechObject.setUtteranceSpeed(0.65); // Set the speaking speed of the TTS
  speechObject.setPitch(1.1); // Set the pitch of the TTS voice
  speechObject.setAccent(US); // Set the Accent of the TTS voice
}
```

loop()

```
void loop()
{
  if( button.isPressed() )
  {
    button.ack();
    // Use updateData() to get the phone to talk!
    speechObject.updateData("Confucius say: Man run in front of car get tired,");
    speechObject.updateData("man run behind car get exhausted.");
    // You will need to break your sentence into multiple lines if your lines are too long.
    // Each speech object can only handle up to 40 characters of text at a time.
    // Punctuation will also affect the way your phone vocalises the text.
  }

  displaybox.update();
  button.update();
  // Do not update the speechObject!

  delay(500); // Always leave a short delay for Bluetooth communication
}
```

Leçon n°10:

Prendre automatiquement des photos avec votre smartphone.

Dans cette leçon, nous allons vous montrer comment utiliser le combo Arduino/Annikken Andee pour configure votre smartphone afin qu'il prenne automatiquement des photos. Nous utiliserons ici un bouton, mais vous pouvez tout aussi bien utiliser un capteur.

Top of the Code

```
#include <SPI.h>
#include <Andee.h>

// We'll just create a display box to give us updates, and a button
// to trigger the camera.
AndeeHelper displaybox;
AndeeHelper button;
AndeeHelper camera; // This object will handle the smartphone camera
```

setup()

```
void setup()
{
  Andee.begin();  // Setup communication between Annikken Andee and Arduino
  Andee.clear();  // Clear the screen of any previous displays
  setInitialData(); // Define object types and their appearance
}
```

setInitialData()

```
void setInitialData()
{
  // Let's draw the button!
  displaybox.setId(0); // Don't forget to assign a unique ID number
  displaybox.setType(DATA_OUT); // Defines object as a display box
  displaybox.setLocation(0,0,FULL);
  displaybox.setTitle("Andee Controlled Camera");
  displaybox.setData("Camera Ready");

  button.setId(1);
  button.setType(BUTTON_IN);
  button.setLocation(1,0,FULL);
  button.setTitle("Take Photo Now");

  // Smartphone Camera Properties //////////////////////////
  camera.setId(2);
  camera.setType(CAMERA); // Set object as a camera

  // You can choose which camera you want to use:
  // - DEFAULT  : default camera
  // - FRONT    : front-facing camera
  // - REAR     : rear-facing camera
  camera.setCamera(DEFAULT);

  // Set whether you wish to enable auto focus before each shot
  // Options available:
  // - ON
  // - OFF
  camera.setAutoFocus(ON); // Options: true or false

  // Set whether you wish to enable flash at each shot
  // Options available:
  // - ON
```

```
  // - OFF
  // - AUTO
  camera.setFlash(ON);

  // Set photo filename prefix. Subsequent photos will
  // have a number appended at the end.
  camera.setPhotoFilename("imageName");

  // If file overwrite is enabled, the same file will be replaced
  // everytime a new photograph is taken. If disabled, a new photo
  // will be generated and stored on the phone.
  // Options available:
  // - ON
  // - OFF
  camera.setFileOverwrite(OFF);

  // Choose storage location to store your photos. The Andee app
  // will create a folder in that location called, "Annikken_Andee"
  // Options available:
  // - DEFAULT : Save in the default photo location on your phone.
  //             If you have set your default photo location as SD Card,
  //             photos taken will be stored on your SD card.
  // - SDCARD  : Save in external SD Card.
  camera.setSaveLocation(DEFAULT);
}
```

loop()

```
void loop()
{
  // You can substitute the button press with a sensor condition instead
  if( button.isPressed() )
  {
    button.ack();
    displaybox.setData("Taking Photo");
    camera.takePhoto(); // Call smartphone camera to take photo

    // Give the smartphone time to complete action. It will take
    // the camera at least 2 seconds to complete the entire
    // photo-taking process. This process may take even longer
    // on slower phones.
    delay(3000);

    // If the phone is still not ready, wait another 2 seconds more
    // before checking again.
    while( !camera.takePhotoResultReady() )
    {
      delay(2000);
    }

    // camera.getLastTakePhotoResult() will produce an error code
    // at the end of the process.
    if(camera.getLastTakePhotoResult() == 0)
    {
      displaybox.setData("Photo Successfully Taken!");
      displaybox.update();
    }
    else
    {
      displaybox.setData("Error!");
      displaybox.update();
      while(true); // Freeze and do not allow user to do anything else
    }
  }

  displaybox.update();
  button.update();

  delay(500); // Always leave a short delay for Bluetooth communication

  displaybox.setData("Camera Ready"); // Reset displaybox message
}
```

Leçon n° 11:

Utiliser le signal Bluetooth pour contrôler des objets.

Voici une autre méthode que vous pouvez utiliser pour contrôler la carte Arduino. Par exemple, vous pouvez utiliser la force du signal Bluetooth pout contrôler des objets ou exécuter des taches, par exemple allumer ou éteindre des lampes dans une pièce simplement en entrant ou en la quittant.

Top of the Code

```
#include <SPI.h>
#include <Andee.h>

// We'll have a displaybox to show you the Bluetooth signal strength
AndeeHelper displaybox;

// We'll just light up an LED to demonstrate this
// LED connected to Pin 2
const int outputPin = 2;

char strBuffer[30];
char signalStr[4];
int signalStrength;
```

setup()

```
void setup()
{
  Andee.begin();  // Setup communication between Annikken Andee and Arduino
  Andee.clear();  // Clear the screen of any previous displays
  setInitialData(); // Define object types and their appearance

  pinMode(outputPin, OUTPUT); // Configures outputPin for output.
  digitalWrite(outputPin, LOW);
}
```

setInitialData()

```
void setInitialData()
{
  displaybox.setId(0);
  displaybox.setType(DATA_OUT);
  displaybox.setLocation(0,0,FULL);
  displaybox.setTitle("Signal Strength");
  displaybox.setUnit("dB");
}
```

loop()

```
void loop()
{
  if( Andee.isConnected() ) // Run only when connected
  {
    // Retrieve Bluetooth information from the Andee and store it in strBuffer
    Andee.sendCommand("GET CONNECTED MAC_ID", strBuffer);
    memcpy(signalStr, &strBuffer[18], 4); // Extract signal strength
    signalStrength = atoi(signalStr); // Convert to int value

    displaybox.setData(signalStrength);
    displaybox.update();

    // We're gonna use a double threshold line to prevent light flickering.
    // If you use a single threshold, when you stand at the edge of the threshold
```

```
    // line, the signal will fluctuate around that value, causing the LED to
    // flicker. This is very undesirable.

    if(signalStrength > -60) // The nearer to zero, the closer you are
    {
      digitalWrite(outputPin, HIGH);
    }

    // Andee will not do anything when you are within the -70db to -60db range.

    if(signalStrength < -70) // When you're very far away
    {
      digitalWrite(outputPin, LOW);
    }

  }
  if( !Andee.isConnected() )
  {
    digitalWrite(outputPin, LOW); // Keep the LED off
  }

  delay(500); // Always leave a short delay for Bluetooth communication
}
```