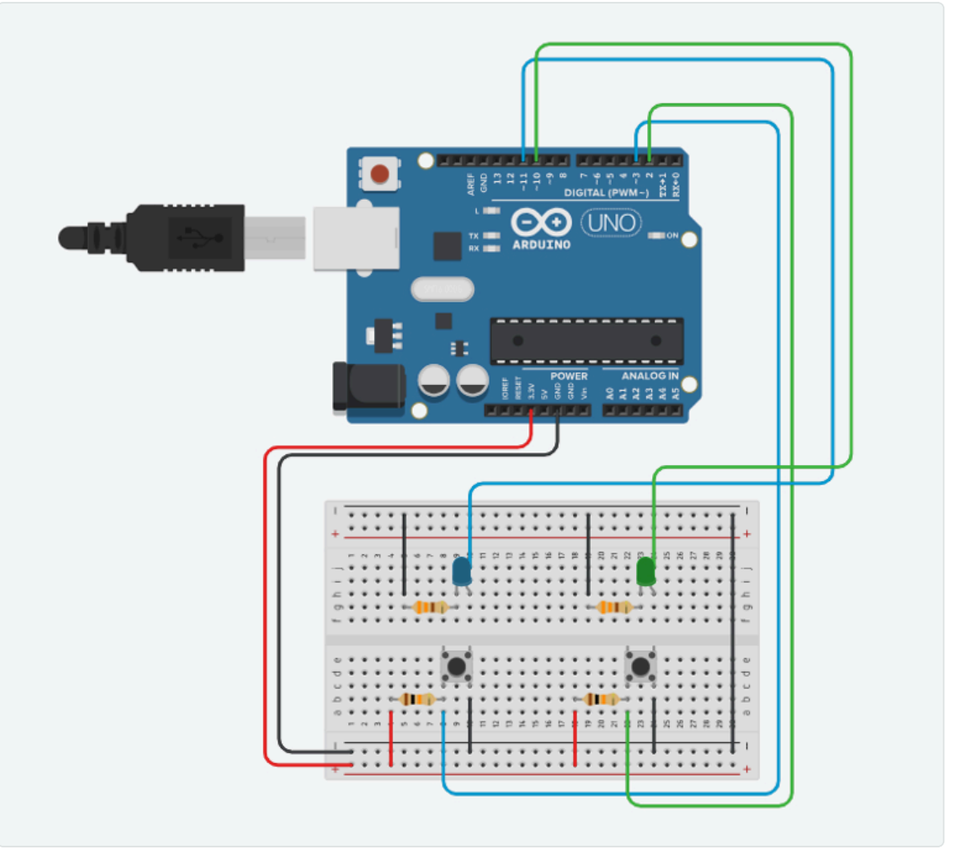
🔌 Arduino to Misty

In this project, you will use Arduino to trigger Misty to change the colour of her chest LED.

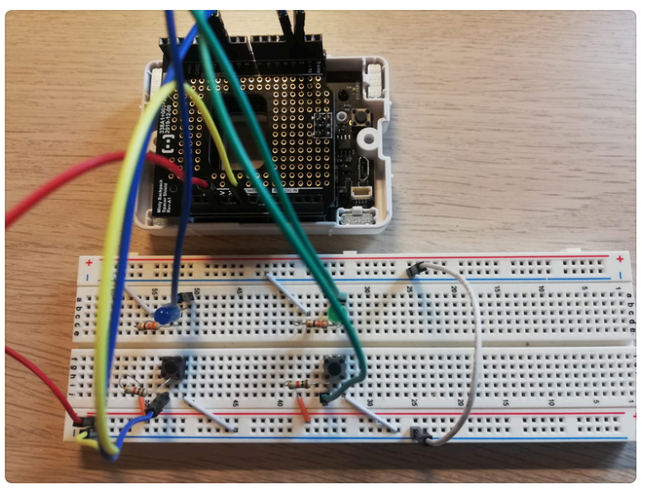
Materials

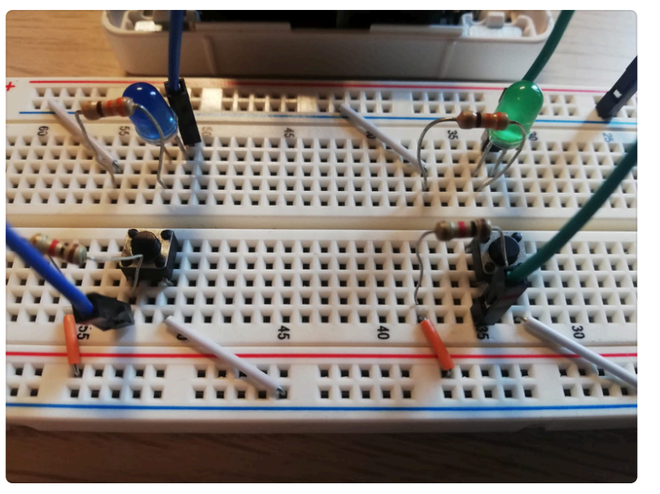
- Misty's Arduino backpack
- 2 buttons
- 2 LEDs
- 2 resistance 330 Ω
- 2 resistance 10 kΩ
- Some wires of different lengths.

Arduino Circuit



It may seem in this scheme that the buttons are not fully connected but in your breadboard, you can close the pins a bit.
You can use the same scheme with Arduino UNO or the arduino Backpack





Arduino Code

```
void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);     //pin green button
  pinMode(3, INPUT_PULLUP);     //pin blue button
  pinMode(10, OUTPUT);          //pin green LED
  pinMode(11, OUTPUT);          //pin blue LED
}

void loop() {
  int pushedg = digitalRead(2); //green button
  int pushedb = digitalRead(3); // blue button

  if (pushedg == LOW){      // green button pressed condition
    digitalWrite(10, HIGH); // green LED on
    Serial.println("green");    // write "green" in the serial
  } else {
    digitalWrite(10, LOW);  // green LED off
  }

  if (pushedb == LOW){      // blue button pressed condition
    digitalWrite(11, HIGH); // blue LED on
    Serial.println("blue"); // write "blue" in the serial
  } else {
    digitalWrite(11, LOW);  // blue LED off
  }
  delay(200);           // wait for 200ms
}
```

The Arduino code is composed of two parts: the void setup and the void loop. The first one will run only once at the beginning and here you set all your variables. The second part will run in a loop until stopped.

In this void setup, you open the serial and set the pins you'll use in this project. In the void loop, you associate the pins to our buttons and then we set two conditions:

If the green button is pressed you'll turn on the green LED on the Arduino board and you will write in the serial the word "green" if this doesn't happen the green LED will be off. It happens the same with the blue button as you can read in the code. At the end, we insert a pause in order not to send too many signals to Misty.

Misty Code

```
from mistyPy.Robot import Robot
from mistyPy.Events import Events

misty = Robot("your_robot_IP_address")

def msg(data):
    print(data)
    if data["message"]["message"] == "green":
misty.change_led(0, 255, 0)
        if data["message"]["message"] == "blue":
misty.change_led(0, 0, 255)

misty.register_event(event_name="serial_message_event", event_type=Events.SerialMessage,
callback_function=msg, keep_alive=True)
misty.keep_alive()
```

To allow Misty to read in the Serial you need to build an event, so you can import the usual libraries that Misty uses to register events. Then we create the robot and the function that will be called with the event.

In this function we read the serial message and if the condition is verified Misty will complete the function. If we press the green button, Arduino will write "green" in the serial and Misty will register "green" as message from the serial port, so she'll act as shown in the function.

Last updated 20 hours ago

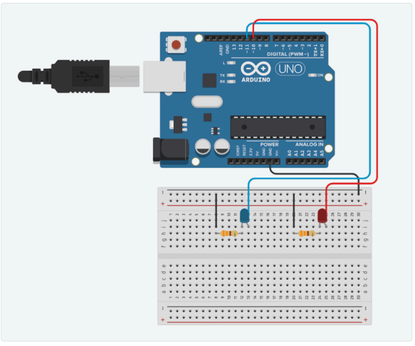
Misty to Arduino

In this project, you'll use MIsty's bump sensor to trigger Arduino to change its LEDs.

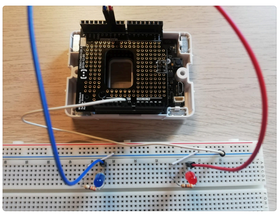
Materials

- Misty's Arduino backpack
- 2 LEDs
- 2 resistance 330 Ω
- Some wires of different lengths.

Arduino Circuit



You can use the same scheme with Arduino UNO or the arduino Backpack



Arduino Code

```
void setup(){
  Serial.begin(9600);
  Serial.setTimeout(100);
  pinMode(10, OUTPUT);          //pin red LED
  pinMode(11, OUTPUT);          //pin blue LED
}

void loop() {
  if (Serial.available() > 0){          // check if something is written in the Serial
    String str = Serial.readStringUntil('\n');  // read the string
    if (str == "red"){                  // compare the string
      digitalWrite(10, HIGH);               // red LED on
      Serial.println("red on");                 // write in the serial monitor
    }
    if (str == "blue"){
      digitalWrite(11, HIGH);
      Serial.println("blue on");
    }
    if (str == "red off"){
      digitalWrite(10, LOW);
      Serial.println("red off");
    }
    if (str == "blue off"){
      digitalWrite(11, LOW);
      Serial.println("blue off");
    }
    if (str == "all off"){
      digitalWrite(10, LOW);
      digitalWrite(11, LOW);
      Serial.println("all off");
    }
    if (str == "all on"){
      digitalWrite(10, HIGH);
      digitalWrite(11, HIGH);
      Serial.println("all on");
    }
  }
delay(500);
}
```

In this code, you can see the same structure as the previous one: A void setup and a void loop. In the void setup, you open the serial, set the maximum milliseconds to wait for serial data (the default is 1000 ms) and define the pins that you'll use in the code.

In the void loop, the first thing you should do is to check if there is something written in the serial. The second step is to read the string that exists in the serial by defining it. To read strings completely you can use the character "\n" at the end, which represents the new line character/enter. Once you get the string the last part is to define all the conditions you can have.

For example :

```
if (str == "red"){
      digitalWrite(10, HIGH);
      Serial.println("red on");
    }
```

If the string you get from Misty is "red" turn on the red LED and write in the Arduino's serial "red on". The Serial.println at the end of each if condition is not necessary for the code but it's helpful for the programmer who can see what's happening in the code from the serial monitor in the Arduino IDE.

Misty Code

```
from mistyPy.Robot import Robot
from mistyPy.Events import Events

misty = Robot("your_robot_IP_address")

def bumped(data):
    print(data)
    if(data["message"]["sensorId"] == "bfr" and data["message"]["isContacted"] == True):
        misty.change_led(255, 0, 0)
        misty.write_serial("red\n")
    if(data["message"]["sensorId"] == "bfl" and data["message"]["isContacted"] == True):
        misty.change_led(0, 0, 255)
        misty.write_serial("blue\n")
    if(data["message"]["sensorId"] == "brr" and data["message"]["isContacted"] == True):
        misty.change_led(0, 255, 0)
        misty.write_serial("red off\n")
    if(data["message"]["sensorId"] == "brl" and data["message"]["isContacted"] == True):
        misty.change_led(255, 127, 0)
        misty.write_serial("blue off\n")

misty.register_event(event_name='bump_event', event_type=Events.BumpSensor, callback_func
misty.keep_alive()
```

To write in the serial you don't need an event, but Misty has it as an element in her Python API. This code is a normal Bump sensor event that associates to each bump sensor a string to write in the serial.

For example:

```
if(data["message"]["sensorId"] == "bfr" and data["message"]["isContacted"] == True):
        misty.change_led(255, 0, 0)
        misty.write_serial("red\n")
```

If the bump front right sensor is contacted turn Misty's chest LED red and write "red\n" in the serial. Arduino will receive the string from the serial, read it until the "\n" character and select the function from its list.

Misty Tracker

This project aims to make Misty a temperature and light tracker in your room!

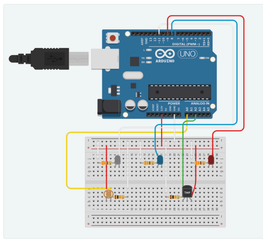
When you are using Arduino to collect data there is a big obstacle: Arduino can't move alone!

Misty will adapt her expression based on the conditions she will be in, for example when it will be dark and cold she will be scared and so on. You can adapt your actions and the path she will drive.

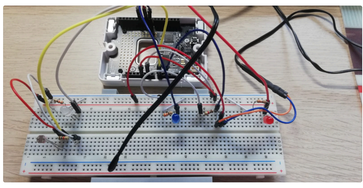
Materials:

- Misty's Arduino Backpack
- Thermistor
- Photoresistor
- 3 LEDs
- 3 resistance 330 Ω
- 2 resistance 10 kΩ
- Some wires of different lenght
- Optional Arduino breedboard support

Arduino Circuit:



Remember to use Misty's Arduino Backpack board, adopt your resistances to your sensors and check your individual connections, for example, our temperature sensor had only two wires.



Arduino Code:

```
// light and temperature
int light_lim = 300;
double temp_lim = 26; // temperature limit

void setup() {
  Serial.begin(9600);
  pinMode(A0, INPUT); // light sensor
  pinMode(A2, INPUT); // temperature sensor
  pinMode(11, OUTPUT); // light LED (white)
  pinMode(9, OUTPUT); // temperature LED hot (red)
  pinMode(8, OUTPUT); // temperature LED cold (blue)
}

void loop() {
  // read the value
  int light_value = analogRead(A0);
  double temp_value = (analogRead(A2) - 32) /1.8; // the value is originally read in Fahr

  //compare the value with limits
  if ((light_value <= light_lim) && (temp_value <= temp_lim)) { // dark and cold
    digitalWrite(11, HIGH);
    digitalWrite(9, LOW);
    digitalWrite(8, HIGH);
    Serial.print("dark and cold|");
  }
  else if ((light_value >= light_lim) && (temp_value <= temp_lim)){ // illuminated and co
    digitalWrite (11, LOW);
    digitalWrite(9, LOW);
    digitalWrite(8, HIGH);
    Serial.print("illuminated and cold|");
  }
  else if ((light_value >= light_lim) && (temp_value >= temp_lim)){ // illuminated and ho
    digitalWrite(11, LOW);
    digitalWrite(9, HIGH);
    digitalWrite(8, LOW);
    Serial.print("illuminated and hot|");
  }
  else if ((light_value <= light_lim) && (temp_value >= temp_lim)){ // dark and hot
    digitalWrite(11, HIGH);
    digitalWrite(9, HIGH);
    digitalWrite(8, LOW);
    Serial.print("dark and hot|");
  }
  else { // in any other case
    digitalWrite(11, LOW);
    digitalWrite(9, LOW);
    digitalWrite(8, LOW);
    Serial.print("searching|");
  }
  String values = "light value: " + String(light_value) + "    temperature value: " + Str
  Serial.println(values);
  delay(2000); // scan every 2 seconds
}
```

First of all, you can set some values of light and temperature that will be the limits to decide if an environment is dark or illuminated and cold or hot.

In this void setup, you open the serial and set the pins you'll use in this project. In this void loop, you read the sensors and write a specific serial message based on your conditions:

There will be four possible conditions, and it has been also added an extra case that will save the program in case none of the conditions is selected.

The Serial message, which is our way of transmitting the information to Misty is built in this way:

`light condition and temperature condition | light value and temperature value`

We insert the | because in this way, in the Misty python code, we can split the message and use them indipendently. We want it because:

- it would be harder to build an if statement with all the different values each time
- you can print the data and only the data that Misty captured in your output

Arduino will scan the different conditions every two seconds.

Misty Code:

```
from mistyPy.Robot import Robot
from mistyPy.Events import Events
import time

ROBOT_IP = "<your_robot_ip_address>"
misty = Robot(ROBOT_IP)

def msg(data):
    conditions = data["message"]["message"].split("|")[0]
    values = data["message"]["message"].split("|")[1]
    print(values)

    if conditions == "dark and cold":
        misty.set_flashlight(True)
        misty.display_image("e_Fear.jpg")
        misty.move_arms(80, 80)
        misty.change_led(0, 0, 255)
    elif conditions == "illuminated and cold":
        misty.set_flashlight(False)
        misty.move_arms(-80, -80)
        misty.change_led(0, 255, 0)
    elif conditions == "illuminated and hot":
        misty.set_flashlight(False)
        misty.play_audio("s_Joy.wav")
        misty.display_image("e_Admiration.jpg")
        misty.change_led(255, 0, 0)
    elif conditions == "dark and hot":
        misty.set_flashlight(True)
        misty.start_action("love")
        misty.change_led(255, 0, 255)
    elif conditions == "searching":
        misty.set_flashlight(False)
        misty.start_action("look-left")
        time.sleep(0.5)
        misty.start_action("look-right")
    else:
        misty.change_led(100, 70, 160)

misty.register_event(event_name="serial_message_event", event_type=Events.SerialMessage,
# my tracked space
for i in range(2):
    misty.drive_time(50, 0, 4000)
    time.sleep(4500)
    for j in range(4):
        misty.move_head(-20, 0, 0)
        time.sleep(1)
        misty.move_head(20, 0, 0)
        time.sleep(1)
    misty.drive_time(0, 50, 6500)
    time.sleep(7000)
    misty.drive_time(50, 0, 4000)
    time.sleep(4500)
    misty.drive_time(0, 50, 6500)
    time.sleep(7000)
# end my tracked space
misty.start_action("hi")
misty.keep_alive()
```

To allow Misty to read in the Serial you need to build an event, so you can import the usual libraries that Misty uses to register events. Then you can create the robot and the function that will be called with the event.

In this function Misty reads the serial message and if the condition that Arduino sent is verified Misty will act according to what you wrote in that case.

In the function, you can split values and conditions so you can use them independently as previously said. In the end, you can loop the space that will be tracked several times, in this case 2 and meanwhile Misty will track all your data.

🌤️ Misty weather forecaster

Have you ever thought that your Misty could become an excellent weather reporter and forecaster?

The Adafruit BME280 board will give you barometric pressure, temperature and humidity!

The humidity level is a nice report to have in your home.
The barometric pressure data are temperature-dependent.
This means that if your temperature goes up Misty will read a lower pressure reading.

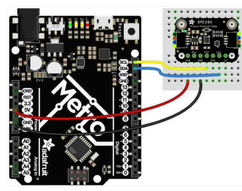
Materials:

- Misty's Arduino Backpack
- Adafruit BME280 board
- Arduino breadboard support

Adafruit BME280 instructions

You can find the instructions for this board in this link

If you're using the breadboard you will need to solder the breadboard 1 pins into the circuitboard.
You can slide the board over the pins and heat the pins, then apply solder. After your board is ready here is the pin layout, this is on an Arduino board but of course, it's the same on Misty backpack. You can use the 3.3volt supply.



Arduino Code:

```
//Libraries
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME280 bme; // I2C
unsigned long delayTime;

// Void Setup to check the initial condition
void setup() {
  Serial.begin(9600);
  while(!Serial); // time to get serial running
  Serial.println(F("BME280 test"));
  unsigned status = bme.begin();
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring, address, sensor I
    while (1) delay(10);
  }
  Serial.println("-- Default Test --");
  delayTime = 1000;
  Serial.println();
}

//Infinite function that reads and sends the data to Misty
void loop() {
  Serial.print("Temperature: ");
  Serial.print(bme.readTemperature());
  Serial.println(" *F");
  Serial.print("Pressure: ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");
  Serial.print("Humidity: ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");
  Serial.println();
  delay(delayTime);
}
```

You can follow these instructions to complete the setup of the BME280 board before starting it.

In this Arduino code, we declare the libraries that we'll use, void setup and void loop.

The void setup prepares the code and runs only once, in this case, we check the presence of the BME280 sensor.

In the void loop, we read and send the data from the Arduino sensor to Misty via serial port.

Misty Code:

```
from mistyPy.Robot import Robot
from mistyPy.Events import Events
import time
import threading

# Connect to Misty
misty = Robot("YOUR MISTY IP")
print("Successfully connected to Misty.")
misty.change_led(0, 255, 0)

#Variables
initial_pressure = None
initial_temperature = None
temperature = None
pressure = None
humidity = None
announce_interval = 360 # 6 minutes in seconds

#Functions
#Normalizing data
misty.speak("Let's see what's going on with your weather")
def parse_data(data):
    message = data["message"]["message"]
    try:
        if "Temperature" in message:
            temp_index = message.index("Temperature: ") + len("Temperature: ")
            temp_end_index = message.index(" *F", temp_index)
            temperature_celsius = float(message[temp_index:temp_end_index].strip())
            temperature_fahrenheit = temperature_celsius * 9.0/5.0 + 32.0
            return ("temperature", round(temperature_fahrenheit, 2))
        elif "Pressure" in message:
            pressure_index = message.index("Pressure: ") + len("Pressure: ")
            pressure_end_index = message.index(" hPa",pressure_index)
            pressure = message[pressure_index:pressure_end_index].strip()
            return ("pressure", pressure)
        elif "Humidity" in message:
            humidity_index = message.index("Humidity: ") + len("Humidity: ")
            humidity_end_index = message.index(" %", humidity_index)
            humidity = message[humidity_index:humidity_end_index].strip()
            return ("humidity", humidity)
    except ValueError:
        return None

#Change Misty's screen
def display_data(temperature, pressure, humidity):
    display_text = f"\nTemperature: {temperature} *F\nPressure:{pressure} hPa\nHumidity:
    misty.display_image("e_SystemBlackScreen.jpg")
    misty.display_text(display_text, "Large")
    print(f"Updated display with: {display_text}")

#Announcement of the weather condition
def announce_weather_condition(pressure):
    if pressure > 974:# change this value according to your experiments
        misty.speak("There is no storm system in the vicinity for sometime and most likel
    elif pressure < 960:
        misty.speak("We are currently within a precipitation system.")

#Starting readings
def announce_readings():
    global temperature, pressure, humidity
    while True: #infinite loop
        #while reading data
        if temperature and pressure and humidity:
            misty.speak(f"The current temperature is {temperature} degrees Fahrenheit, th
            announce_weather_condition(pressure)
        time.sleep(announce_interval)

#Pressure variation
def check_pressure_change(new_pressure):
    global initial_pressure
    if initial_pressure is None:
        initial_pressure = new_pressure
        pressure_difference = new_pressure - initial_pressure
        return pressure_difference
    if abs(pressure_difference) >= 5:
        direction = "up" if pressure_difference > 0 else "down"
        misty.speak(f"I am detecting a distinct change in pressure by 5 points {direction}

#Read the data
def serial_msg_test(data):
    global temperature, pressure, humidity, initial_temperature, initial_pressure
    print(f"Received data: {data}")
    parsed = parse_data(data)
    if parsed:
        if parsed[0] == "temperature":
            temperature = parsed[1]
            if initial_temperature is None:
                initial_temperature = temperature
                misty.speak(f"The current temperature is {initial_temperature} degrees Fa
        elif parsed[0] == "pressure":
            pressure = float(parsed[1])
            if initial_pressure is None:
                initial_pressure = pressure
                misty.speak(f"The current pressure is {initial_pressure} hPa.")
                announce_weather_condition(pressure)
            check_pressure_change(pressure)
        elif parsed[0] == "humidity":
            humidity = parsed[1]

    if temperature and pressure and humidity:
        display_data(temperature, pressure, humidity)

# Start the announcement thread
announcement_thread = threading.Thread(target=announce_readings)
announcement_thread.daemon = True
announcement_thread.start()

# Register serial event
misty.register_event(event_name="serial_message_event", event_type=Events.SerialMessage,
print("Serial event registered. Waiting for data...")

# Keep the program running
# CHOOSE OF THE FOLLOWING SOLUTION

#you can use this variation suggested by
try:
    while True:
        pass # No delay to ensure real-time updates
except KeyboardInterrupt:
    print("Program terminated by user.")
except Exception as e:
    print(f"Error in keep_alive: {e}")

#or the usual misty command
misty.keep_alive()
```

This is a very well-built code. On top, we have the libraries that Misty will use. Then we connect with our robot via IP address as usual and declare the variables that we'll use in the code.

Then we have the functions that we'll use to read and normalize the data, check the variation in pressure and change Misty's display and behaviour according to the data.

In the last lines, we find the function that registers the event and two possible solutions to keep Misty alive. Remember to use only one of the two solutions.

Notes from the creator:

The barometric values I entered for when Misty says "a storm is en route or not ..." will change depending on where you live and the altitude you are at.

The best way to figure out your own is to run the program frequently, on sunny clear days, rainy days and most importantly note the levels during a major storm (snow, thunderstorms etc).

Eventually, you will figure out which values are appropriate to your own location.

Thanks

A special thanks to Scott who shared with us his Misty weather forecaster project!
You can find his contacts here if you are willing to collaborate with him on new Misty projects!
crmfghtr@yahoo.com

Last updated 6 days ago

📺 Conference Assistant

This project aims to make Misty a conference assistant.

This is a basic interpretation but her code can be enriched with many actions and sentences that can assist the speaker at a conference. This code will allow the speaker to control the presentation triggering Misty's bump sensors. If the rear right sensor is triggered Misty will move to the next slide, if the rear left sensor is triggered Misty will move to the previous slide. After you run the Python code from your computer open your presentation and wait for the magic to happen.

We have also added another command: if you press the front left sensor all the events will be stopped and your program can finish under your control.

We will utilize the Python `pyautogui` library for this purpose, which enables our code to control the keyboard. To install pyautogui, use the following command line in your terminal: "`pip install pyautogui`". Press enter and wait for the installation of all the required packages.

It can only be used in a desktop environment and is not compatible with the Misty Studio Python Interface, so ensure you have all the necessary resources for the task before initiating.

```
from mistyPy.Robot import Robot
from mistyPy.Events import Events
import pyautogui
import time

misty = Robot("YOUR_IP_ADDRESS") #your Robot IP Address

time.sleep(2)
def bumped(event):
    if(event["message"]["sensorId"] == "brr" and event["message"]["isContacted"] ==
True):
        misty.change_led(0, 255, 255)
        pyautogui.press("right")
        print("next")

    if(event["message"]["sensorId"] == "brl" and event["message"]["isContacted"] ==
True):
        misty.change_led(255, 0, 255)
        pyautogui.press('left')
        print("previous")

    if(event["message"]["sensorId"] == "bfl" and event["message"]["isContacted"] ==
True):
        misty.unregister_all_events()
        print("finished")

misty.register_event(event_name='bump_event', event_type=Events.BumpSensor,
callback_function=bumped, keep_alive=True)
misty.keep_alive()
```

Last updated 24 days ago

🚨 Misty Intruder Alert

In this project you can learn how to have Misty trigger an intruder alert if she doesn't recognize a person and send a message to your phone.

If Misty detects a face that's not stored in her memory, she will access your WhatsApp web and send a text message to your phone number saying: "Intruder!". If she does recognize the person, she will send the message "person_name is home!". Before executing the code, open https://web.whatsapp.com/ and log in with your phone number.

We will utilize the Python `pywhatkit` library for this purpose, which enables message sending on WhatsApp in a semi-automatic way. To install pywhatkit, use the following command line in your terminal: "`pip install pywhatkit`". Press enter and wait for the installation of all the required packages.

Another required action prior to using Whatsapp is to train Misty on the faces you want her to recognize as friends (misty studio Explore>Vision>Train faces) and upload all the files you'll run in the code, both images and audio files (misty studio Explore>Expressions>Upload audio files or Upload images or videos).

Note: Sometimes the WhatsApp message might not be sent before the website closes. In that case, try running the code again.

To reset Misty after running the code, you can access its Misty Studio and in the wizard section, click on the "Body Reset" action.

It can only be used in a desktop environment and is not compatible with the Misty Studio Python Interface, so before initiating, ensure you have all the necessary resources for the task.

```
from mistyPy.Robot import Robot
from mistyPy.Events import Events
import time
import pywhatkit
from datetime import datetime

misty = Robot("YOUR_IP_ADDRESS") #your Robot IP Address

def send_whatsapp_message(MyNumber, MyTextMessage): #function to send a message on WhatsA
    myobj = datetime.now()
    pywhatkit.sendwhatmsg(MyNumber, MyTextMessage, myobj.hour, myobj.minute + 1, 10, True

MyNumber = "___" #your phone number (don't forget to include the country code)

misty.start_face_recognition()

def recognized(data):
    print(data)
    if data["message"]["label"] == 'unknown person': #intruder
        misty.display_image("e_Anger.jpg")
        misty.transition_led(255, 0, 0, 0, 0, 255, "Blink", 100)
        misty.move_arms(-80, -80)
        misty.speak("Intruder! Intruder!")
        misty.play_audio("Police Siren Sound Effect.mp3", 50)
        MyTextMessage = "Intruder!"
        send_whatsapp_message(MyNumber, MyTextMessage)
    else :
        misty.display_image("e_Joy2.jpg") #familiar face
        misty.change_led(0, 255, 0)
        for i in range (2):
            misty.move_arms(80, -80, 50, 50)
            time.sleep(1)
            misty.move_arms(80, 0, 50, 50)
            time.sleep(1)
        MyTextMessage = data["message"]["label"] + " is home!"
        send_whatsapp_message(MyNumber, MyTextMessage)


misty.register_event(event_name='face_recognition_event', event_type=Events.FaceRecogniti
misty.keep_alive()
```

You can use this link to access the Police Siren Sound Effect.mp3 file https://youtu.be/HKieGUH9pzg

Last updated 3 months ago

MistyGPT

Welcome to MistyGPT! Using this project template you can learn how to integrate ChatGPT in your Misty to have autonomous conversations and combine it with Langchain to give Misty verbal commands as well as ask for information.

Note: Can only be used in a desktop environment, not compatible with the Misty Studio Python Interface.

MistyGPT.py | Constants.py | Data.txt

```
import os
import sys
from mistyPy.Robot import Robot
from mistyPy.Events import Events
import time
import random
import requests

# Initialize Misty with your robot's IP address
misty = Robot("YOUR_IP_ADDRESS")
#Robot default
misty.set_default_volume(50)

#To install dependencies use pip install langchain_community,  pip install openai
and pip install constants
import constants
from langchain_community.document_loaders import TextLoader
from langchain.indexes import VectorstoreIndexCreator
from langchain_community.llms import OpenAI
from langchain_community.chat_models import ChatOpenAI

os.environ["OPENAI_API_KEY"] = constants.APIKEY
print("API key loaded")

#query = sys.argv[1]

loader = TextLoader('Python-SDK-main\data.txt')
print("data.txt is loaded")
#loader = DirectoryLoader(".", glob="*.txt")
index = VectorstoreIndexCreator().from_loaders([loader])

def speech_captured(data):
    if data["message"]["step"] == "CompletedASR":
        user_input = data["message"]["text"]
        process_user_input(user_input)
        print(user_input)


def process_user_input(user_input):
    mistyOutput = index.query(user_input, llm=ChatOpenAI())
    #print(mistyOutput)
    moveArms = "move my arms"
    moveHead = "move my head"
    moveForward = "go forward"
    moveBackward = "go backward"
    moveForGesture1 = "intelligence"
    lowerVolume = "lower my volume"
    higherVolume = "higher my volume"
    changeDisplay= "change my display"
    print(mistyOutput)
    misty.speak_and_listen(mistyOutput)
    if moveForGesture1 in mistyOutput:
        misty.move_arms(-70, 50, 40, 40)
        time.sleep(1)
        print("left arm moved")
        misty.move_arms(50, 50, 40, 40)
    elif moveArms in mistyOutput:
        misty.move_arms(-50, -50, 40, 40)
        time.sleep(2)
        misty.move_arms(50, 50, 40, 40)
        print("arms moved")
    elif moveHead in mistyOutput:
        misty.move_head(0, -25, 0, 100, None, None)
        time.sleep(2)
        misty.move_head(0, 25, 0, 100, None, None)
        time.sleep(2)
        misty.move_head(0, 0, 0, 100, None, None)
        print("arms moved")
    elif moveForward in mistyOutput:
        misty.drive_time(5000,1,5000,0)
        print("moving forward")
    elif moveBackward in mistyOutput:
        misty.drive_time(-5000,1,5000,0)
        print("moving forward")
    elif lowerVolume in mistyOutput:
        misty.set_default_volume(50)
    elif higherVolume in mistyOutput:
        misty.set_default_volume(100)
    elif changeDisplay in mistyOutput:
        misty.display_image("e_JoyGoofy3.jpg")
        time.sleep(3)
        misty.display_image("e_EcstacyHilarious.jpg")
        time.sleep(3)
        misty.display_image("e_defaultcontent.jpg")

def recognized(data):
    print(data)
    misty.speak("Yay, Hi " + data["message"]["label"], 1)
    misty.stop_face_recognition()
    time.sleep(2)
    misty.start_dialog()
    misty.speak_and_listen("How can I help you today", utteranceId="required-for-
callback")

#If Misty is lifted she gets a bit touchy about that.
def touch_sensor(data):
    if data["message"]["sensorId"] == "cap" and data["message"]["isContacted"] ==
True:
        touched_sensor = data["message"]["sensorPosition"]
        print(touched_sensor)
        if touched_sensor == "Scruff":
            misty.play_audio("s_Rage.wav")
            misty.display_image("e_Anger.jpg")
            time.sleep(3)
           #Triggers face recognition event to initate ChatGPT
        if touched_sensor == "HeadFront":
            misty.move_head( -5, 0, 0, 85, None, None)
            misty.display_image("e_Joy2.jpg")
            misty.speak("Aha")
            time.sleep(1)
            misty.start_face_recognition()
            #Stops ChatGPT event
        if touched_sensor == "Chin":
            misty.move_head(0, -50, 0, 150, None, None)
            misty.play_audio("s_Love.wav")
            misty.display_image("e_Love.jpg")
            time.sleep(2)
            misty.display_image("e_DefaultContent.jpg")
            misty.unregister_event("arbitrary-name")

misty.register_event(event_name="touch-sensor",
                     event_type=Events.TouchSensor,
                     callback_function=touch_sensor,
                     keep_alive=True)


misty.register_event(event_name="arbitrary-name",
                     event_type=Events.DialogAction,
                     callback_function=speech_captured,
                     keep_alive=True)

misty.register_event(event_name='face_recognition_event',
                     event_type=Events.FaceRecognition,
                     callback_function=recognized,
                     keep_alive=False)

#misty.speak(index.query(query, llm=ChatOpenAI()))

x = 4
while (x > 3):
    misty.display_image("e_DefaultContent.jpg")
    misty.move_arms(30, 30, 40, 40)
    misty.move_head(0, 0, 0, 85, None, None)
    time.sleep(5)
    misty.display_image("e_ContentLeft.jpg")
    time.sleep(3)
    misty.move_arms(20, 10, 40, 40)
    time.sleep(2)
    misty.move_head(0, -10, 0, 60, None, None)
    time.sleep(5)
    misty.display_image("e_ContentRight.jpg")
    time.sleep(3)
    misty.move_head(0, 10, 0, 60, None, None)
    time.sleep(5)
    misty.move_arms(10, 20, 40, 40)


print("testing")
misty.keep_alive()
```

Last updated 4 months ago

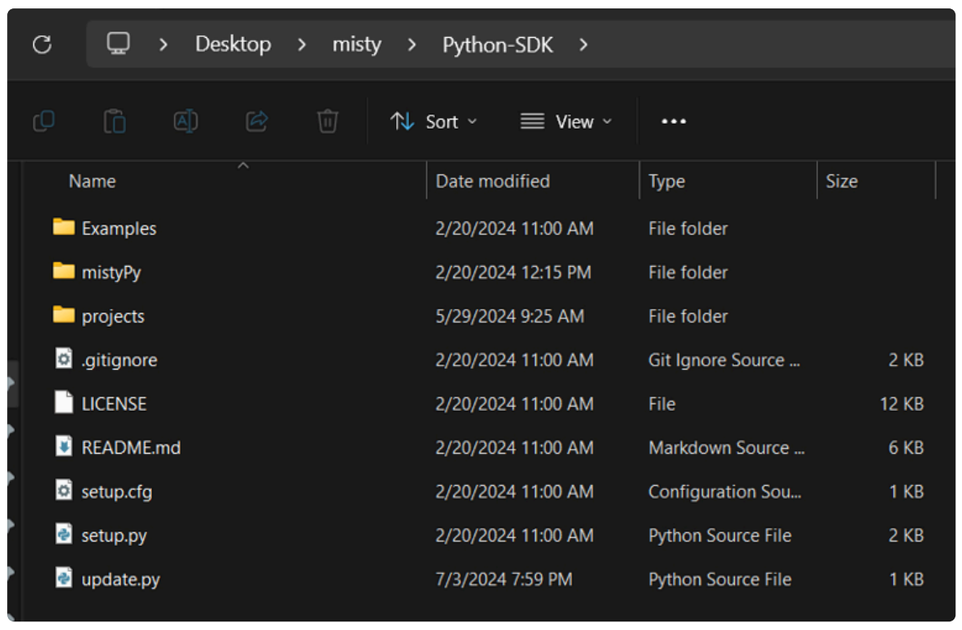
🏷️ QR code detector

With this feature it's possible for Misty to detect and read QR codes. These can be used to tag spaces, for example: labs, classrooms, kitchens or living rooms to make Misty start actions.

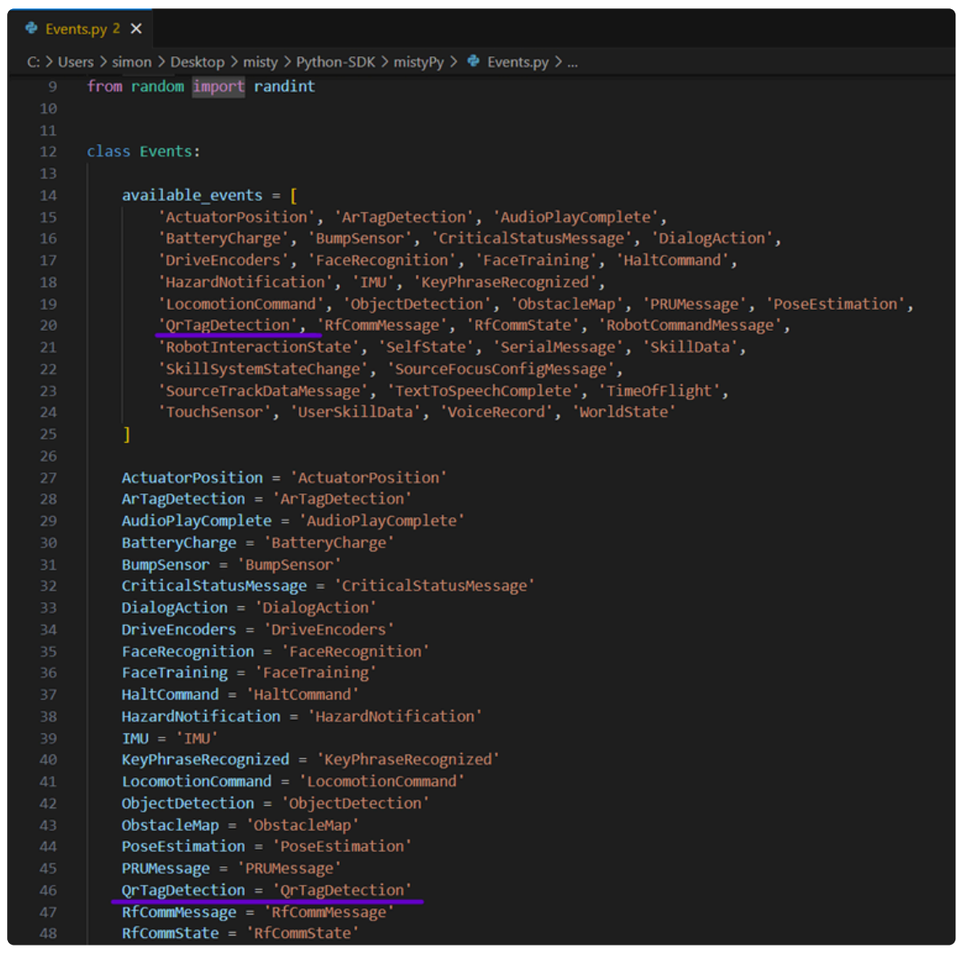
We suggest to use a printed QR code, about 15x15 cm. The QR code that we used for this project is the Misty Lessons one.

This project will only work in the Misty Desktop Environment because we will modify the events file.

Open the folder containing the Python-SDK that you use for the desktop environment.
The folder should look like this one.



Open the folder MistyPy, open the file Events.py in Visual Studio code and modify the Events class to add the QRdetection Event.



Now you're ready to use Misty's scanning QR capabilities in Python.

Python Code

```python
from mistyPy.Robot import Robot
from mistyPy.Events import Events

misty = Robot("YOUR_IP_ADDRESS") #your Robot IP Address

misty.change_led(0, 255, 0)
misty.move_head(0, 0, 0)

misty.start_qr_tag_detector()

def code(data):
    print(data) #this shows all the data related to the event
    print(data['message']['decodedInfo'])

misty.register_event(event_name='qrtag_event', event_type=Events.QrTagDetection, callback
misty.keep_alive()
```

In the first part of the code, we set the robot by initializing the libraries, calling it and setting it in a standard position.
Once these actions are done we start the event like any other event in the Misty world by calling the action *.start_qr_tag_detector().*

We create a function that will handle the data read by scanning the QR code. We can then register the event and then keep it alive.

You can insert if statements to handle different QR codes and assign to each a different Misty action.

For example, if we want Misty to turn her RGB LED purple when she recognizes the QR code of the Misty lessons or have the RGB LED red when she doesn't, our function can look like this one:

```python
def code(data):
    print(data['message']['decodedInfo'])
    if data['message']['decodedInfo'] == 'unknown person': #intruder
        misty.change_led(100, 70, 160)
    else :
        misty.change_led(255, 0, 0)
```

Last updated 24 days ago