**DATA COLLECTION**

The data collection process for audio recording on the RedBoard Artemis ATP through the PDM interface can be summarized in the following steps:

1. Set up the hardware. Connect the RedBoard Artemis ATP to your computer and install the AmbiqSuite-R2.3.2 SDK with the SparkFun board BSP files included.
2. Upload the firmware to the RedBoard Artemis ATP. Copy the pdm\_fft example code from the AmbiqSuite-R2.3.2/boards\_sfe/common/examples/pdm\_fft folder to your Arduino IDE and upload it to the RedBoard Artemis ATP.
3. Run the PC-side data collector. The Python data collector program can be found in the GitHub repository for this project. Run the program and select the serial port that the RedBoard Artemis ATP is connected to.
4. Trigger a recording. Type the letter r in the Python data collector program and press enter. The RedBoard Artemis ATP will start recording audio for approximately 1 second.
5. Receive the recorded data. Once the recording is finished, the RedBoard Artemis ATP will send the recorded data to the PC-side data collector program. The Python data collector program will save the recorded data to a file.
6. To generate sparks, you can connect a relay to the Arduino Due board. The Python data collector program will send a command to the Arduino Due board to turn on the relay before starting the recording. The relay will then generate sparks for the duration of the recording.

Step-by-step guide for collecting data:

1. Connect the RedBoard Artemis ATP to your computer and install the AmbiqSuite-R2.3.2 SDK with the SparkFun board BSP files included.
2. Copy the pdm\_fft example code from the AmbiqSuite-R2.3.2/boards\_sfe/common/examples/pdm\_fft folder to your Arduino IDE and upload it to the RedBoard Artemis ATP.
3. Run the Python data collector program and select the serial port that the RedBoard Artemis ATP is connected to.
4. Start playing a randomly chosen audio file on a loudspeaker.
5. Type the letter r in the Python data collector program and press enter.
6. The RedBoard Artemis ATP will start recording audio for approximately 1 second.
7. The Arduino Due board will turn on the relay, generating sparks.
8. Once the recording is finished, the RedBoard Artemis ATP will send the recorded data to the PC-side data collector program.
9. The Python data collector program will save the recorded data to a file and turn off the relay.
10. Repeat steps 4-9 to collect more data.

Once you have collected enough data, you can use it to train and evaluate your machine learning model.

**DATA PREPARATION**

Once you have collected a dataset of spark sounds and background noises, you need to preprocess the data to ensure that it is in a format that can be used by the neural network model. This may involve the following steps:

1. Normalize the data: This involves scaling the data to a specific range, such as 0 to 1 or -1 to 1. This is important to ensure that the neural network model does not give too much weight to any one feature.
2. Convert the data to a spectrogram: A spectrogram is a visual representation of the frequency content of a sound signal. It can be generated using a Fourier transform. Spectrograms are often used in sound detection tasks because they can be used to capture the unique characteristics of different sound events.
3. Resize the data: The neural network model must be trained on inputs of a specific size. Therefore, you may need to resize the spectrograms to ensure that they are all the same size.

In addition to the above steps, you may also want to perform other data preprocessing operations, such as:

1. Filter the data: You can use filters to remove noise or other unwanted signals from the data.
2. Augment the data: Data augmentation is a technique that can be used to increase the size and diversity of the dataset. This can be done by creating new data samples from existing data samples by applying transformations such as cropping, flipping, and adding noise.

Once you have pre-processed the data, you can split it into training, validation, and test sets. The training set will be used to train the neural network model, the validation set will be used to evaluate the model during training, and the test set will be used to evaluate the model after training is complete.

Here are some additional tips for preparing the data:

1. Use a consistent preprocessing pipeline. It is important to use a consistent preprocessing pipeline for all of the data. This will help to ensure that the data is consistent and that the neural network model is trained on a fair representation of the data.
2. Use a variety of data augmentation techniques. Data augmentation can be a very effective way to improve the performance of the neural network model. Try using a variety of different data augmentation techniques to see how they affect the performance of the model.
3. Validate the data. It is important to validate the data before training the neural network model. This will help to identify any errors or inconsistencies in the data.

**MODEL TRAINING**

To train the model, you can use the following steps:

1. Choose a neural network architecture. There are a variety of different neural network architectures that can be used for sound detection. Some popular architectures include convolutional neural networks (CNNs) and recurrent neural networks (RNNs). CNNs are particularly well-suited for modelling sequential data, such as audio signals. RNNs are also well-suited for modelling sequential data, but they can be more computationally expensive than CNNs.
2. Implement the neural network model. Once you have chosen a neural network architecture, you need to implement the model in a programming language such as Python or TensorFlow.
3. Compile the neural network model. Once you have implemented the neural network model, you need to compile it into a format that can be used to train the model.
4. Train the neural network model. To train the neural network model, you need to feed it the pre-processed data and the corresponding labels. The model will learn to predict the labels for the data.
5. Evaluate the neural network model. Once the neural network model is trained, you need to evaluate its performance on a held-out test set. This will help you to identify any areas where the model needs improvement.

Here are some additional tips for training the model:

1. Use a small learning rate. A learning rate is a hyperparameter that controls how much the weights of the neural network model are updated during training. A small learning rate will help to prevent the model from overfitting to the training data.
2. Use a regularization technique. Regularization is a technique that can help to prevent overfitting. Some popular regularization techniques include L1 regularization and L2 regularization.
3. Use a validation set. A validation set is a set of data that is used to evaluate the performance of the model during training. This will help you to identify any areas where the model needs improvement.
4. Train the model for a sufficient number of epochs. An epoch is one pass through the entire training set. It is important to train the model for a sufficient number of epochs to ensure that it has converged.

Once you have trained the neural network model, you can deploy it to the RedBoard Artemis ATP module.

**MODEL EVALUATION**

To evaluate a machine learning model, you will need to split your dataset into two sets: a training set and a test set. The training set is used to train the model, and the test set is used to evaluate the model's performance on unseen data.

Once you have split your dataset, you can train your model on the training set. Once the model is trained, you can evaluate its performance on the test set. To do this, you will need to feed the test data to the model and calculate the evaluation metrics.

If the model's performance on the test set is satisfactory, then you can deploy the model to production. However, if the model's performance on the test set is not satisfactory, then you may need to retrain the model with more data or with a different model architecture.

Here are some additional tips for evaluating a machine learning model:

1. Use a large and diverse test set to ensure that the model is generalizing well to unseen data.
2. Use multiple evaluation metrics to get a more complete picture of the model's performance.
3. Evaluate the model on different subsets of the test set to identify any potential biases.
4. Compare the model's performance to the performance of other models that have been trained on the same task.

**MODEL DEPLOYMENT**

To deploy your model to the board, you will need to convert the model to a format that is compatible with the board's hardware and software. This may involve converting the model to a smaller format, such as TensorFlow Lite, or compiling the model to native code.

Once the model is in a compatible format, you can upload it to the board using a variety of methods, such as using a USB cable or a serial connection.

Once the model is uploaded to the board, you can run it using the board's hardware and software. This may involve writing a simple program to load the model and make predictions.