This Python code serves as a robust "Industrial Monitoring, Management, and Data Acquisition System" designed to operate on a Raspberry Pi. Its primary objective is to enhance industrial operations by incorporating several critical functionalities. The code imports a range of essential libraries, including RPi.GPIO for GPIO control, SimpleMFRC522 for RFID card operations, Adafruit\_DHT for collecting data from a DHT11 sensor (for temperature and humidity), Pushbullet for sending real-time alerts, thingspeak for data logging, and other libraries for handling CSV files and timestamps.

The code configures two GPIO (General Purpose Input/Output) pins, namely 'led' (connected to GPIO pin 21) and 'buzzer' (connected to GPIO pin 20). These GPIO pins serve as indicators to provide visual (LED) and auditory (buzzer) alerts as necessary during system operation. Additionally, it establishes the type of sensor in use (DHT11) and specifies the GPIO pin (GPIO 26) to which the sensor is connected for real-time monitoring of temperature and humidity. The code also requires the user to input ThingSpeak channel information, including the channel ID, as well as the write and read API keys. ThingSpeak is an IoT platform utilized for data logging and remote data access, which plays a pivotal role in this system.

Furthermore, the code initializes the Pushbullet service with a unique access token, enabling the system to send real-time alerts to designated devices. The system's RFID capabilities are provided by the SimpleMFRC522 module. It defines a timer variable and proceeds to create two primary functions, 'func1()' and 'measure(channel)'. 'func1()' handles RFID card scans and access control, using the RFID reader to detect the card's unique ID. If a valid card is presented, the system triggers a Pushbullet notification indicating "Valid RFID!" and subsequently proceeds to monitor temperature and humidity using the 'measure()' function. The data collection is repeated five times, each separated by a 2-second delay. In the case of an invalid RFID card, the system sends an "Invalid RFID!" alert using Pushbullet and activates both the LED and buzzer for 5 seconds, indicating an intruder alert. The code maintains a timer to ensure that the alert persists for the specified duration. Should an error occur during this process, the code prints "Connection failed" to the console.

The 'measure(channel)' function is responsible for capturing temperature and humidity data using the DHT11 sensor. It conducts five successive measurements with 2-second intervals and logs the readings. Timestamps for each data point are generated using the datetime module, capturing both the date and time of each measurement. The data is then sent to the ThingSpeak channel using the 'channel.update()' method, mapping temperature to field 1 and humidity to field 2. Simultaneously, the code retrieves and displays the latest data from the ThingSpeak channel using the 'channel.get()' method.

Finally, to ensure long-term data record-keeping, the code appends the collected data to a CSV file called "sensor\_data.csv." The CSV file is opened in append mode to continually add new data points while preserving previous records. The file is equipped with column headers ('Date', 'Time', 'Temperature', 'Humidity'), and these headers are written to the file if it is empty (i.e., on first use). Subsequently, the data is appended as rows of dictionaries in the CSV file. In the event of an error during data measurement, the code prints an error message for diagnostic purposes.

The main execution block commences the operation of the system. It initializes the ThingSpeak channel and enters an infinite loop, continually awaiting RFID card scans. For each RFID card scan, the 'func1()' function is executed, followed by a 2-second pause. This loop ensures that the system continuously functions, providing comprehensive industrial monitoring, security, and data acquisition capabilities.