Things already added/modified in the base code, so the automation module can work:

- Added __init.py__ to code folder, so this could be recognised by python as a module/package and not a directory (before, I can’t import IPS class from main.py)
- Renamed code folder to src, as from code import.. generated errors, sometimes python can confuse this name to an already existing module in default environments
- In ips_protocol/recording2_pb2.py the last 4 imports were: from ips_protocol import… It was changed with: from src.ips_protocol import… When the IPS class was imported from another file, an old import raised NoModuleFoundError. This says in the first comments that it's an auto generated class, so this could be a bug in the way the file was generated.

Code Review:

- All 3 methods can be splitted into smaller methods, so this code can be easily extended, reduce future duplicate code and it’s easier to debug and understand. A new structure could look something like:
    - read_recording
        - read from proto()
        - populate dataframes()
        - add magnetics positions()
        - eliminate future records()
    - magnetics_pos_calc

      ```
      … #same code until here
      for i in self.positions.index[:-1]:
          speed_x,speed_y = calculate_speed()
          while tj<self.positions.loc[i+1][‘t’]:
              calculate_amplitude()
              row +=1
              check_magnetic_in_cell()
              tj = self.magnetics.loc[row][‘t’]
      return None
      ```
    - set_rect_grid
        - construct_grid_shape()
        - compute_grid()

        - plot_grid()

- Grid is closed when the program is terminated, and the plot is right before it's finished. We should not close the plot even if the program is terminating. Instead of imshow you can use plt.plot()
- read_recoring docstring for magnetics dataframe structures has not the 'x' and 'y' after 'accuracy'
- In the read_recording method, there's no need to iterate through measurements.positions and magnetics and also hardcode the columns. Pandas has a special method that can turn a record into a dataframe (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.from_records.html) You can use
  self.positions = pd.DataFrame.from_records(measurements.positions)
  self.magnetics = pd.DataFrame.from_records(measurements.amgnetics)
- In magnetics_pos_cal method, if the speed is assumed to be constant, you don't need to calculate the speed for every 2 consecutive ground truth points. speed_x and speed_y can be calculated once outside the for statement
- In set_rect_grid, x and y axis could be calculated faster. You don't need an array of points and get it's len. It should be enough to make for example:
  x_axis = np.ceil((int(np.ceil(self.positions.x.max())-int(np.floor(self.positions.x.min()))/self.cell_size[0])+1. For y_axis it's same formula, but with y values and cell_size[1]
- x_min and y_min could be calculated before the x_axis and y_axis, and also you could add x_max and y_max, and substitute them in the previous formula, so it'll be easy to digest it.