Supplementary File

Table S1 Factorial Design for the Selection of Best Factor

Std	Run	Factor 1 A: Temperature ºC	Factor 2 B: pH	Factor 3 C: Substrate Concentration g/L	Factor 4 D: Cell Loading OD600 Value	Factor 5 E: Time hr	Factor 6 F: Air Flow Rate L/min	Factor 7 G: Mass of Immobilized bead g	Response 1 Ethanol Yield %
18	1	40	4.5	50	1	96	0.5	50	25.806
21	2	25	4.5	200	1	96	0.5	50	41.9679
1	3	25	4.5	50	1	24	2.5	50	35.8326
25	4	25	4.5	50	5	96	0.5	10	32.0943
28	5	40	5.5	50	5	96	0.5	10	19.6758
5	6	25	4.5	200	1	24	0.5	10	19.6758
7	7	25	5.5	200	1	24	2.5	50	16.9269
11	8	25	5.5	50	5	24	2.5	10	39.5964
31	9	25	5.5	200	5	96	0.5	10	32.1606
20	10	40	5.5	50	1	96	2.5	10	35.1594
12	11	40	5.5	50	5	24	0.5	50	33.3132
4	12	40	5.5	50	1	24	2.5	50	30.3297
26	13	40	4.5	50	5	96	2.5	50	37.8063
13	14	25	4.5	200	5	24	2.5	10	28.4478
14	15	40	4.5	200	5	24	0.5	50	24.3831
19	16	25	5.5	50	1	96	0.5	50	31.3089
30	17	40	4.5	200	5	96	0.5	10	40.4175
32	18	40	5.5	200	5	96	2.5	50	42.5697
24	19	40	5.5	200	1	96	0.5	50	35.6388
8	20	40	5.5	200	1	24	0.5	10	21.7311
27	21	25	5.5	50	5	96	2.5	50	39.2292
22	22	40	4.5	200	1	96	2.5	10	38.4744
23	23	25	5.5	200	1	96	2.5	10	32.7267
15	24	25	5.5	200	5	24	0.5	50	24.8013
2	25	40	4.5	50	1	24	0.5	10	29.9472
16	26	40	5.5	200	5	24	2.5	10	40.9836
17	27	25	4.5	50	1	96	2.5	10	27.1779
3	28	25	5.5	50	1	24	0.5	10	21.2823
29	29	25	4.5	200	5	96	2.5	50	43.4112
9	30	25	4.5	50	5	24	0.5	50	30.6714
10	31	40	4.5	50	5	24	2.5	10	34.3995
6	32	40	4.5	200	1	24	2.5	50	28.3968

Table S2: Central Composite design (CDD) of Experiment for Ethanol Yield Optimization.

		Factor 1	Factor 2	Factor 3	Factor 4	Response 1
Std	Run	A:Time	B:Air Flow Rate	C:Cell Loading	D:Mass of Bead	Ethanol Yield
		hr	L/min	OD600	g	%
27	1	60	1.5	3	30	47.4
6	2	96	0.5	5	10	45.8833
17	3	-12	1.5	3	30	45.1667
28	4	60	1.5	3	30	46.7333
1	5	24	0.5	1	10	44.2417
3	6	24	2.5	1	10	44.2333
9	7	24	0.5	1	50	44.3667
25	8	60	1.5	3	30	46.4
12	9	96	2.5	1	50	45.5
5	10	24	0.5	5	10	44.7667
29	11	60	1.5	3	30	46.4
14	12	96	0.5	5	50	45.65
24	13	60	1.5	3	70	46.7
16	14	96	2.5	5	50	47.9167
8	15	96	2.5	5	10	44.5
10	16	96	0.5	1	50	44.3333
7	17	24	2.5	5	10	45.2333
23	18	60	1.5	3	-10	45.3667
18	19	132	1.5	3	30	45.1
22	20	60	1.5	7	30	45.65
26	21	60	1.5	3	30	47.4
21	22	60	1.5	-1	30	42.9167
20	23	60	3.5	3	30	45.3
13	24	24	0.5	5	50	45.2833
19	25	60	-0.5	3	30	44.8733
2	26	96	0.5	1	10	44.3142
11	27	24	2.5	1	50	45.6611
30	28	60	1.5	3	30	46.7889
4	29	96	2.5	1	10	43.6833
15	30	24	2.5	5	50	46.4

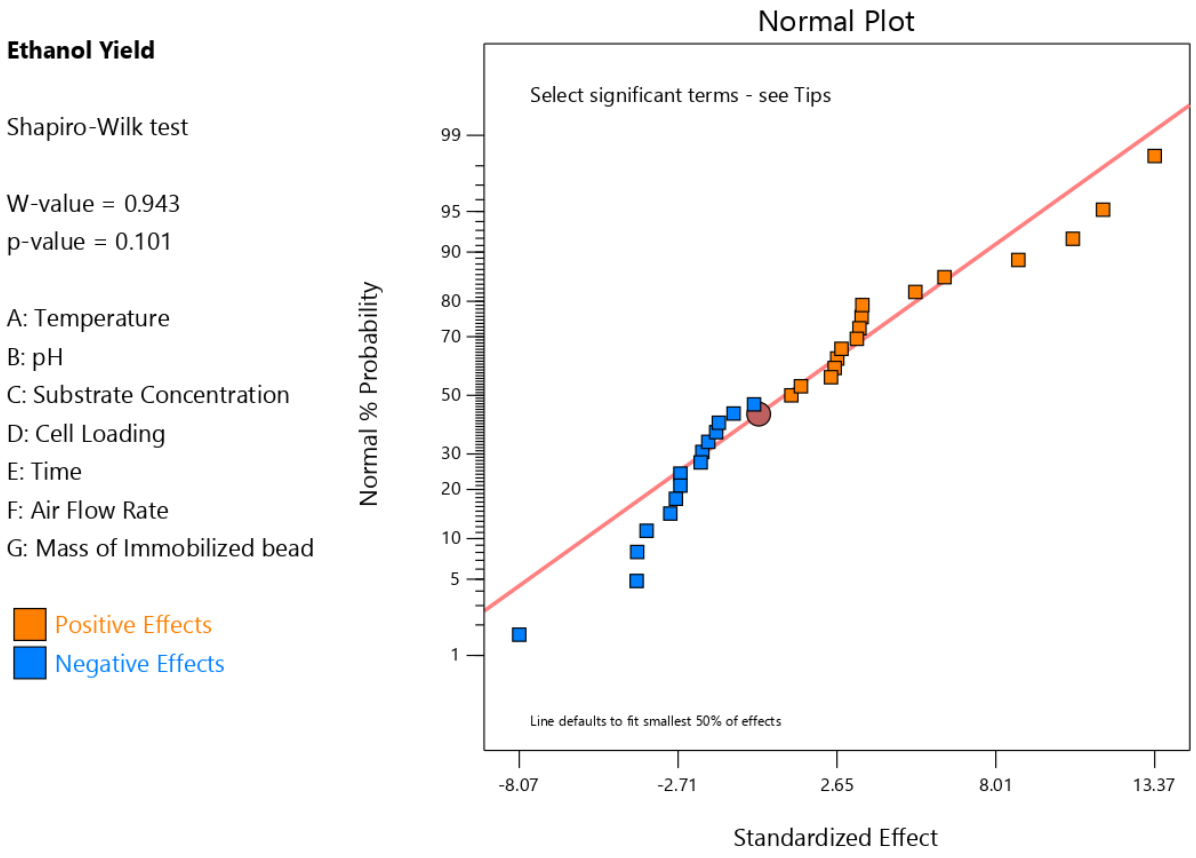

Figure S1 Normality Plot of Ethanol Yield

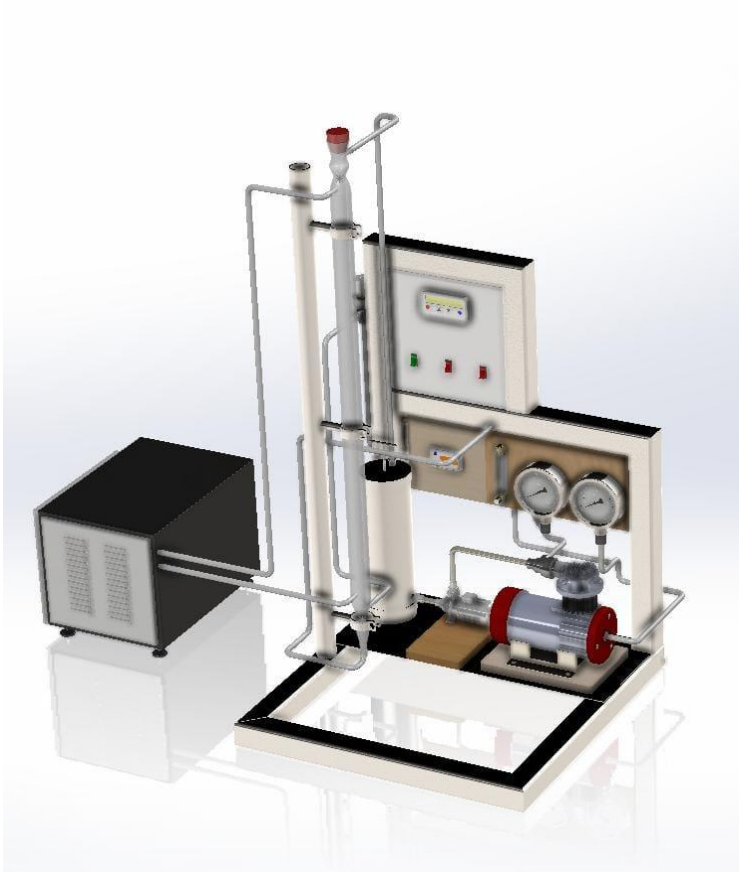

Figure S2: Schematic diagram of the bubble column bioreactor

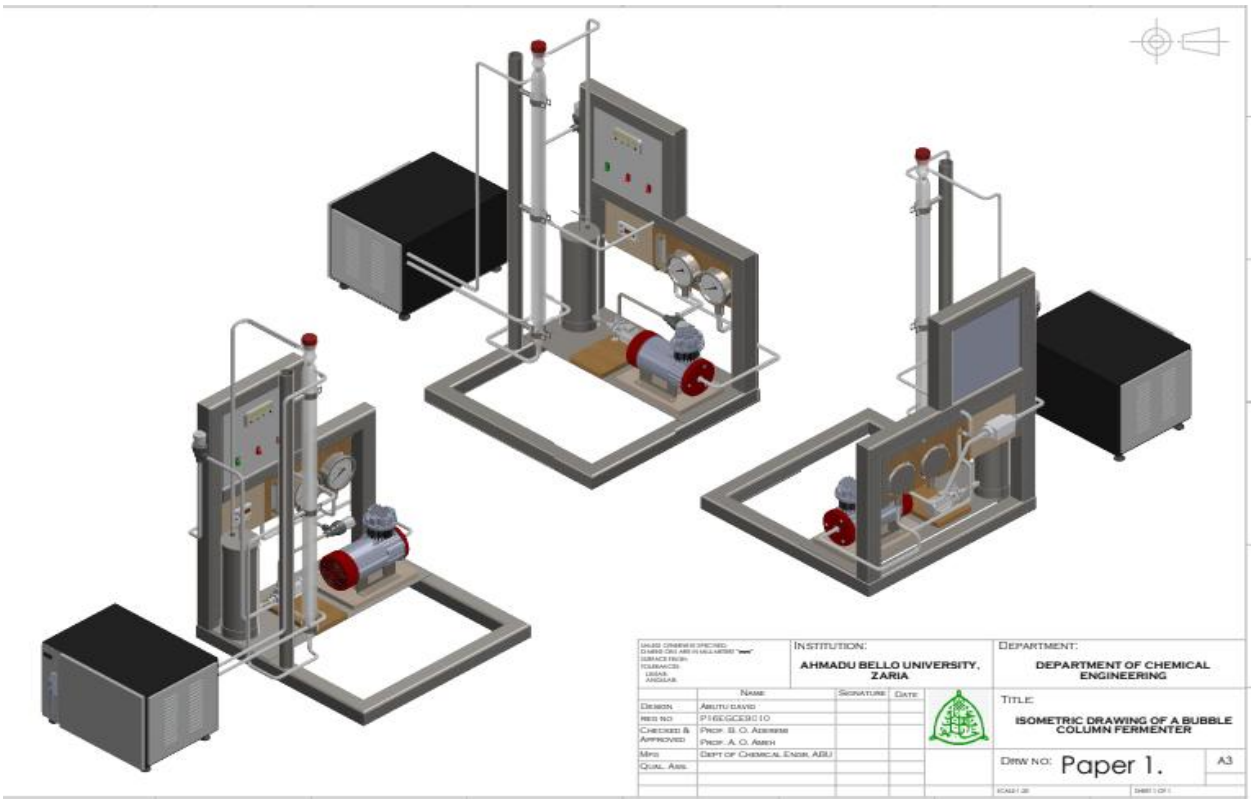

Figure S3 Schematic diagram of the bubble column bioreactor

Figure S4 Picture of the Experimental Setup

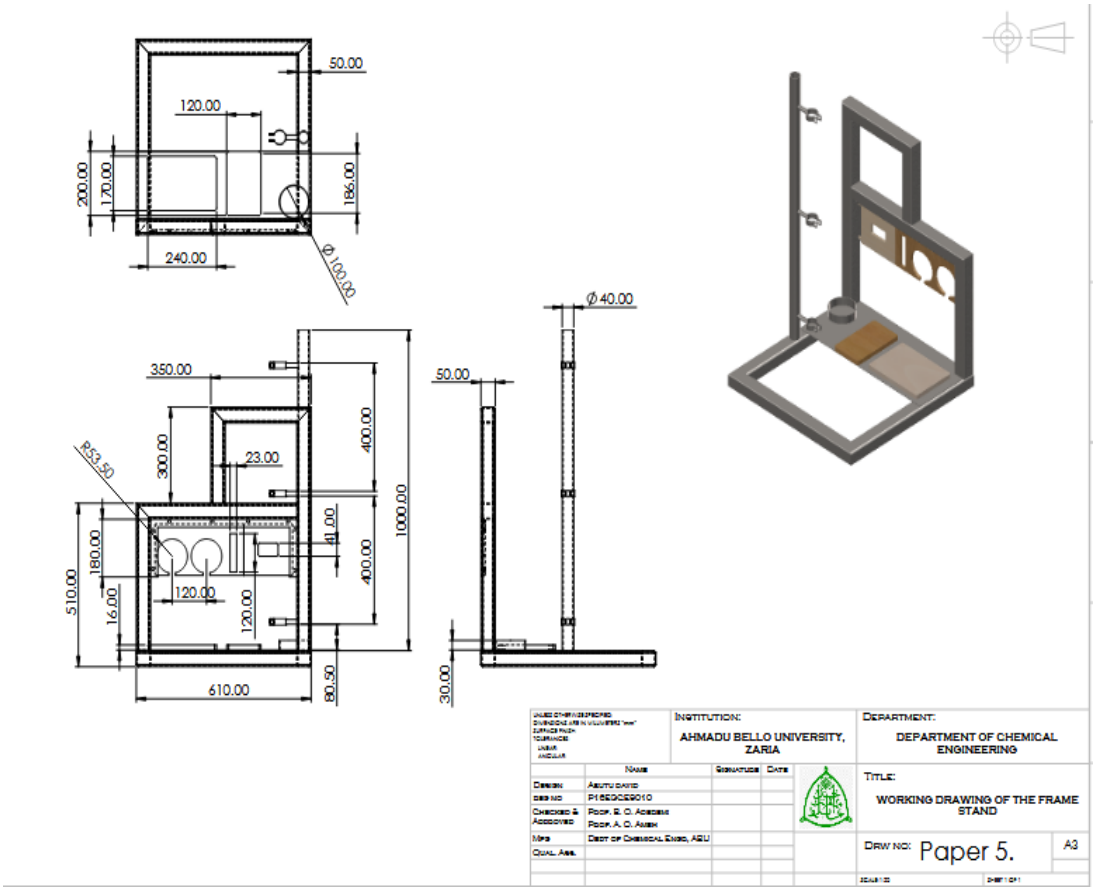

Figure S5 Working Drawing of the Experimental Setup

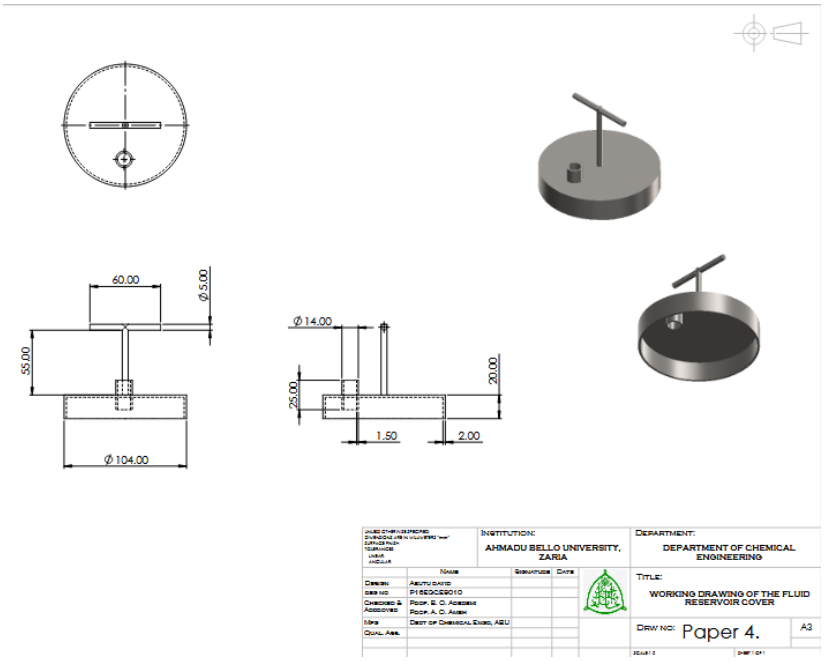

Figure S6 Working Drawing of the Experimental Setup

S- Python Codes Run with Google Co-Lab

 # Google Colab Python Program to Calculate Reactor Volume, Aeration Power, Glucose Depletion, Biomass Growth, Ethanol Production, and CO_2 Evolution

```
import numpy as np
import matplotlib.pyplot as plt

# Given Parameters
Y_O2_X = 1.1  # Oxygen yield coefficient on biomass (g O2/g cells)
Y_X_S = 0.5   # Biomass yield on glucose (g cells/g glucose)
r_S = 2       # Glucose consumption rate (g glucose/L·h)
k_La = 50     # Oxygen mass transfer coefficient (1/h)
epsilon_g = 0.15  # Gas holdup fraction (15%)
P_O2 = 0.21   # Partial pressure of oxygen in air (atm)
H = 769.2     # Henry's constant for oxygen (atm·L/mol)
C_O2 = 0.005  # Dissolved oxygen concentration (g/L)

# Ethanol and CO2 Production Parameters
Y_P_S = 0.05  # Ethanol yield on glucose (g ethanol/g glucose)
Y_CO2_S = 1.0 # CO2 yield on glucose (g CO2/g glucose)

# Solve for required oxygen flow rate (F_O2)
F_O2 = 0.001785 / 60000  # Convert L/min to m³/s

# Calculate reactor volume (V)
V = (F_O2 * Y_O2_X * Y_X_S * r_S) / (k_La * (1 - epsilon_g) * (P_O2 / H - C_O2))

# Aeration Power Calculation
rho_g = 1.225  # Air density (kg/m³ at 25°C)
g = 9.81  # Gravitational acceleration (m/s²)
H_t = 0.3  # Assumed liquid column height in reactor (m)
eta = 0.5  # Aeration efficiency (50%)

P_aeration = (F_O2 * rho_g * g * H_t) / eta  # Power in Watts

# Glucose Depletion Over Time
S0 = 50  # Initial glucose concentration (g/L)
time = np.arange(0, 25, 1)  # Time in hours (0 to 24 hours)
S_t = S0 * np.exp(-r_S * time)  # Glucose concentration over time
S_consumed = S0 - S_t  # Glucose consumed over time

# Biomass Growth Over Time
X0 = 0.1  # Initial biomass concentration (g/L)
mu = 0.3  # Specific growth rate (h^-1)
X_t = X0 * np.exp(mu * time)  # Biomass concentration over time

# Ethanol Production Over Time
P_E = Y_P_S * S_consumed  # Ethanol concentration over time

# CO2 Evolution Over Time
CO2_t = Y_CO2_S * S_consumed  # CO2 concentration over time

# Display reactor volume and power requirement
print(f'Calculated Reactor Volume: {V * 1000:.3f} mL')  # Convert to mL
print(f'Required Aeration Power: {P_aeration:.3f} W')

# Plot Results
fig, axs = plt.subplots(3, 1, figsize=(8, 12))
```

```
# Glucose Depletion
axs[0].plot(time, S_t, 'b', linewidth=2)
axs[0].set_xlabel('Time (hours)')
axs[0].set_ylabel('Glucose (g/L)')
axs[0].set_title('Glucose Depletion Over Time')
axs[0].grid(True)

# Biomass Growth
axs[1].plot(time, X_t, 'r', linewidth=2)
axs[1].set_xlabel('Time (hours)')
axs[1].set_ylabel('Biomass (g/L)')
axs[1].set_title('Biomass Growth Over Time')
axs[1].grid(True)

# Ethanol and CO2 Evolution
axs[2].plot(time, P_E, 'g', linewidth=2, label='Ethanol')
axs[2].plot(time, CO2_t, 'k', linewidth=2, label='CO2')
axs[2].set_xlabel('Time (hours)')
axs[2].set_ylabel('Concentration (g/L)')
axs[2].set_title('Ethanol and CO2 Production Over Time')
axs[2].legend()
axs[2].grid(True)

plt.tight_layout()
plt.show()
```

 # Google Colab Python Program to Calculate Key Fermentation Parameters in a Bubble Column Fermenter

```
import numpy as np
import matplotlib.pyplot as plt

# Given Parameters
Y_O2_X = 1.1  # Oxygen yield coefficient on biomass (g O2/g cells)
Y_X_S = 0.5   # Biomass yield on glucose (g cells/g glucose)
r_S = 2       # Glucose consumption rate (g glucose/L·h)
k_La = 50     # Oxygen mass transfer coefficient (1/h)
epsilon_g = 0.15  # Gas holdup fraction (15%)
P_O2 = 0.21   # Partial pressure of oxygen in air (atm)
H = 769.2     # Henry's constant for oxygen (atm·L/mol)
C_O2 = 0.005  # Dissolved oxygen concentration (g/L)
C_O2_min = 0.002  # Minimum oxygen level to maintain

# Ethanol and CO2 Production Parameters
Y_P_S = 0.05  # Ethanol yield on glucose (g ethanol/g glucose)
Y_CO2_S = 1.0 # CO2 yield on glucose (g CO2/g glucose)

# Solve for initial oxygen flow rate (F_O2)
F_O2 = 0.001785  # L/min (assumed)
F_O2 = F_O2 / 60000  # Convert L/min to m³/s

# Reactor Volume Calculation
V = (F_O2 * Y_O2_X * Y_X_S * r_S) / (k_La * (1 - epsilon_g) * (P_O2 / H - C_O2))

# Time Scale for Fermentation
time = np.arange(0, 49, 1)  # 0 to 48 hours

# Initial Conditions
S0 = 50  # Initial glucose concentration (g/L)
X0 = 0.1  # Initial biomass concentration (g/L)
mu = 0.3  # Specific growth rate (h^-1)

# Preallocate Arrays
```

```
S_t = np.zeros_like(time, dtype=float)
X_t = np.zeros_like(time, dtype=float)
OTR_t = np.zeros_like(time, dtype=float)
OUR_t = np.zeros_like(time, dtype=float)
F_O2_t = np.zeros_like(time, dtype=float)

# Set Initial Values
S_t[0] = S0
X_t[0] = X0
F_O2_t[0] = F_O2  # Initial aeration rate

# Loop for Dynamic Aeration & Fermentation
for t in range(1, len(time)):
    # Update Biomass Growth
    X_t[t] = X_t[t-1] * np.exp(mu * 1)  # Biomass grows exponentially

    # Update Glucose Consumption
    S_t[t] = S_t[t-1] * np.exp(-r_S * 1)  # First-order glucose depletion

    # Oxygen Uptake Rate (OUR)
    OUR_t[t] = Y_O2_X * X_t[t]

    # Oxygen Transfer Rate (OTR)
    OTR_t[t] = k_La * (P_O2 / H - C_O2)

    # Dynamic Oxygen Flow Rate Adjustment
    if OUR_t[t] > OTR_t[t]:  # If oxygen demand exceeds supply
        F_O2_t[t] = F_O2_t[t-1] * 1.1  # Increase aeration by 10%
    elif C_O2 < C_O2_min:  # If dissolved oxygen is too low
        F_O2_t[t] = F_O2_t[t-1] * 1.2  # Increase aeration by 20%
    else:
        F_O2_t[t] = F_O2_t[t-1]  # Maintain same aeration

    # Prevent Excessive Aeration
    if F_O2_t[t] > 0.01:  # Max limit (10 mL/min)
        F_O2_t[t] = 0.01

# Find Optimal Fermentation Time (t_opt) for Peak Biomass
t_opt = np.argmax(X_t)

# Display Results
print(f'Calculated Reactor Volume: {V * 1000:.3f} mL')
print(f'Optimal Fermentation Time: {t_opt} hours')
print(f'Final Oxygen Flow Rate: {F_O2_t[-1] * 60000:.6f} L/min')

# Plot Results
fig, axs = plt.subplots(3, 2, figsize=(12, 10))

# Glucose Depletion
axs[0, 0].plot(time, S_t, 'b', linewidth=2)
axs[0, 0].set_xlabel('Time (hours)')
axs[0, 0].set_ylabel('Glucose (g/L)')
axs[0, 0].set_title('Glucose Depletion Over Time')
axs[0, 0].grid()

# Biomass Growth
axs[0, 1].plot(time, X_t, 'r', linewidth=2)
axs[0, 1].set_xlabel('Time (hours)')
axs[0, 1].set_ylabel('Biomass (g/L)')
axs[0, 1].set_title('Biomass Growth Over Time')
axs[0, 1].grid()
```

```
# Oxygen Uptake Rate (OUR)
axs[1, 0].plot(time, OUR_t, 'm', linewidth=2)
axs[1, 0].set_xlabel('Time (hours)')
axs[1, 0].set_ylabel('OUR (g O2/L·h)')
axs[1, 0].set_title('Oxygen Uptake Rate (OUR)')
axs[1, 0].grid()

# Oxygen Transfer Rate (OTR)
axs[1, 1].plot(time, OTR_t, 'c', linewidth=2)
axs[1, 1].set_xlabel('Time (hours)')
axs[1, 1].set_ylabel('OTR (g O2/L·h)')
axs[1, 1].set_title('Oxygen Transfer Rate (OTR)')
axs[1, 1].grid()

# Dynamic Oxygen Flow Rate
axs[2, 0].plot(time, F_O2_t * 60000, 'k', linewidth=2)
axs[2, 0].set_xlabel('Time (hours)')
axs[2, 0].set_ylabel('Oxygen Flow (L/min)')
axs[2, 0].set_title('Dynamic Oxygen Flow Rate')
axs[2, 0].grid()

# Highlight Optimal Fermentation Time
axs[2, 1].plot(time, X_t, 'r', linewidth=2)
axs[2, 1].plot(t_opt, X_t[t_opt], 'ko', markersize=8, markerfacecolor='g')
axs[2, 1].set_xlabel('Time (hours)')
axs[2, 1].set_ylabel('Biomass (g/L)')
axs[2, 1].set_title('Optimal Fermentation Time')
axs[2, 1].grid()
axs[2, 1].legend(['Biomass', 'Peak Biomass Time'])

plt.tight_layout()
plt.show()
```

 # Google Colab Python Program for Optimized Fermentation and Dynamic Aeration Control

```
import numpy as np
import matplotlib.pyplot as plt

# Given Parameters
Y_O2_X = 1.1  # Oxygen yield coefficient on biomass (g O2/g cells)
Y_X_S = 0.5   # Biomass yield on glucose (g cells/g glucose)
r_S = 2       # Glucose consumption rate (g glucose/L·h)
k_La = 50     # Oxygen mass transfer coefficient (1/h)
epsilon_g = 0.15  # Gas holdup fraction (15%)
P_O2 = 0.21   # Partial pressure of oxygen in air (atm)
H = 769.2     # Henry's constant for oxygen (atm·L/mol)
C_O2 = 0.005  # Dissolved oxygen concentration (g/L)
C_O2_min = 0.002  # Minimum oxygen level to maintain

# Ethanol and CO2 Production Parameters
Y_P_S = 0.05  # Ethanol yield on glucose (g ethanol/g glucose)
Y_CO2_S = 1.0 # CO2 yield on glucose (g CO2/g glucose)

# Solve for initial oxygen flow rate (F_O2)
F_O2 = 0.001785  # L/min (assumed)
F_O2 = F_O2 / 60000  # Convert L/min to m³/s

# Reactor Volume Calculation
V = (F_O2 * Y_O2_X * Y_X_S * r_S) / (k_La * (1 - epsilon_g) * (P_O2 / H - C_O2))

# Time Scale for Fermentation
time = np.arange(0, 49, 1)  # 0 to 48 hours
```

```
# Initial Conditions
S0 = 50  # Initial glucose concentration (g/L)
X0 = 0.1  # Initial biomass concentration (g/L)
mu = 0.3  # Specific growth rate (h^-1)

# Preallocate Arrays
S_t = np.zeros_like(time, dtype=float)
X_t = np.zeros_like(time, dtype=float)
OTR_t = np.zeros_like(time, dtype=float)
OUR_t = np.zeros_like(time, dtype=float)
F_O2_t = np.zeros_like(time, dtype=float)

# Set Initial Values
S_t[0] = S0
X_t[0] = X0
F_O2_t[0] = F_O2  # Initial aeration rate

# Loop for Dynamic Aeration & Fermentation
for t in range(1, len(time)):
    # Update Biomass Growth
    X_t[t] = X_t[t-1] * np.exp(mu * 1)  # Biomass grows exponentially

    # Update Glucose Consumption
    S_t[t] = S_t[t-1] * np.exp(-r_S * 1)  # First-order glucose depletion

    # Oxygen Uptake Rate (OUR)
    OUR_t[t] = Y_O2_X * X_t[t]

    # Oxygen Transfer Rate (OTR)
    OTR_t[t] = k_La * (P_O2 / H - C_O2)

    # Dynamic Oxygen Flow Rate Adjustment
    if OUR_t[t] > OTR_t[t]:  # If oxygen demand exceeds supply
        F_O2_t[t] = F_O2_t[t-1] * 1.1  # Increase aeration by 10%
    elif C_O2 < C_O2_min:  # If dissolved oxygen is too low
        F_O2_t[t] = F_O2_t[t-1] * 1.2  # Increase aeration by 20%
    else:
        F_O2_t[t] = F_O2_t[t-1]  # Maintain same aeration

    # Prevent Excessive Aeration
    if F_O2_t[t] > 0.01:  # Max limit (10 mL/min)
        F_O2_t[t] = 0.01

# Find Optimal Fermentation Time (t_opt) for Peak Biomass
t_opt = np.argmax(X_t)

# Display Results
print(f'Calculated Reactor Volume: {V * 1000:.3f} mL')
print(f'Optimal Fermentation Time: {t_opt} hours')
print(f'Final Oxygen Flow Rate: {F_O2_t[-1] * 60000:.6f} L/min')

# Plot Results
fig, axs = plt.subplots(3, 2, figsize=(12, 10))

# Glucose Depletion
axs[0, 0].plot(time, S_t, 'b', linewidth=2)
axs[0, 0].set_xlabel('Time (hours)')
axs[0, 0].set_ylabel('Glucose (g/L)')
axs[0, 0].set_title('Glucose Depletion Over Time')
axs[0, 0].grid()

# Biomass Growth
```

```
axs[0, 1].plot(time, X_t, 'r', linewidth=2)
axs[0, 1].set_xlabel('Time (hours)')
axs[0, 1].set_ylabel('Biomass (g/L)')
axs[0, 1].set_title('Biomass Growth Over Time')
axs[0, 1].grid()

# Oxygen Uptake Rate (OUR)
axs[1, 0].plot(time, OUR_t, 'm', linewidth=2)
axs[1, 0].set_xlabel('Time (hours)')
axs[1, 0].set_ylabel('OUR (g O2/L·h)')
axs[1, 0].set_title('Oxygen Uptake Rate (OUR)')
axs[1, 0].grid()

# Oxygen Transfer Rate (OTR)
axs[1, 1].plot(time, OTR_t, 'c', linewidth=2)
axs[1, 1].set_xlabel('Time (hours)')
axs[1, 1].set_ylabel('OTR (g O2/L·h)')
axs[1, 1].set_title('Oxygen Transfer Rate (OTR)')
axs[1, 1].grid()

# Dynamic Oxygen Flow Rate
axs[2, 0].plot(time, F_O2_t * 60000, 'k', linewidth=2)
axs[2, 0].set_xlabel('Time (hours)')
axs[2, 0].set_ylabel('Oxygen Flow (L/min)')
axs[2, 0].set_title('Dynamic Oxygen Flow Rate')
axs[2, 0].grid()

# Highlight Optimal Fermentation Time
axs[2, 1].plot(time, X_t, 'r', linewidth=2)
axs[2, 1].plot(t_opt, X_t[t_opt], 'ko', markersize=8, markerfacecolor='g')
axs[2, 1].set_xlabel('Time (hours)')
axs[2, 1].set_ylabel('Biomass (g/L)')
axs[2, 1].set_title('Optimal Fermentation Time')
axs[2, 1].grid()
axs[2, 1].legend(['Biomass', 'Peak Biomass Time'])

plt.tight_layout()
plt.show()
```

 # Google Colab Python Program for Optimized Fermentation with pH Control and Dynamic Aeration

```
import numpy as np
import matplotlib.pyplot as plt

# Given Parameters
Y_O2_X = 1.1  # Oxygen yield coefficient on biomass (g O2/g cells)
Y_X_S = 0.5   # Biomass yield on glucose (g cells/g glucose)
r_S = 2       # Glucose consumption rate (g glucose/L·h)
k_La = 50     # Oxygen mass transfer coefficient (1/h)
epsilon_g = 0.15  # Gas holdup fraction (15%)
P_O2 = 0.21   # Partial pressure of oxygen in air (atm)
H = 769.2     # Henry's constant for oxygen (atm·L/mol)
C_O2 = 0.005  # Dissolved oxygen concentration (g/L)
C_O2_min = 0.002  # Minimum oxygen level to maintain

# Ethanol and CO2 Production Parameters
Y_P_S = 0.05  # Ethanol yield on glucose (g ethanol/g glucose)
Y_CO2_S = 1.0 # CO2 yield on glucose (g CO2/g glucose)

# pH Control Parameters
pH_0 = 5.5   # Initial pH
pH_min = 5.0 # Minimum acceptable pH
k_CO2 = 0.05 # CO2 influence on pH drop (empirical factor)
```

```
NaOH_added = np.zeros(49)  # Buffer addition tracking (g/L)

# Solve for initial oxygen flow rate (F_O2)
F_O2 = 0.001785 / 60000  # Convert L/min to m³/s

# Reactor Volume Calculation
V = (F_O2 * Y_O2_X * Y_X_S * r_S) / (k_La * (1 - epsilon_g) * (P_O2 / H - C_O2))

# Time Scale for Fermentation
time = np.arange(0, 49)  # Extended time (0 to 48 hours for optimization)

# Initial Conditions
S0 = 50  # Initial glucose concentration (g/L)
X0 = 0.1  # Initial biomass concentration (g/L)
mu = 0.3  # Specific growth rate (h^-1)

# Preallocate Arrays
S_t = np.zeros_like(time, dtype=float)
X_t = np.zeros_like(time, dtype=float)
OTR_t = np.zeros_like(time, dtype=float)
OUR_t = np.zeros_like(time, dtype=float)
F_O2_t = np.zeros_like(time, dtype=float)
pH_t = np.zeros_like(time, dtype=float)
CO2_t = np.zeros_like(time, dtype=float)

# Set Initial Values
S_t[0] = S0
X_t[0] = X0
pH_t[0] = pH_0
F_O2_t[0] = F_O2  # Initial aeration rate

# Loop for Dynamic Aeration, pH Control & Fermentation
for t in range(1, len(time)):
    # Update Biomass Growth
    X_t[t] = X_t[t-1] * np.exp(mu * 1)

    # Update Glucose Consumption
    S_t[t] = S_t[t-1] * np.exp(-r_S * 1)

    # CO2 Evolution Over Time
    CO2_t[t] = Y_CO2_S * (S0 - S_t[t])

    # Oxygen Uptake Rate (OUR)
    OUR_t[t] = Y_O2_X * X_t[t]

    # Oxygen Transfer Rate (OTR)
    OTR_t[t] = k_La * (P_O2 / H - C_O2)

    # Dynamic Oxygen Flow Rate Adjustment
    if OUR_t[t] > OTR_t[t]:  # If oxygen demand exceeds supply
        F_O2_t[t] = F_O2_t[t-1] * 1.1  # Increase aeration by 10%
    elif C_O2 < C_O2_min:  # If dissolved oxygen is too low
        F_O2_t[t] = F_O2_t[t-1] * 1.2  # Increase aeration by 20%
    else:
        F_O2_t[t] = F_O2_t[t-1]  # Maintain same aeration

    # Prevent Excessive Aeration
    if F_O2_t[t] > 0.01:  # Max limit (10 mL/min)
        F_O2_t[t] = 0.01

    # pH Drop Due to CO2 Production
    pH_t[t] = pH_0 - k_CO2 * CO2_t[t]
```

```
    # pH Correction with NaOH if Below Threshold
    if pH_t[t] < pH_min:
        NaOH_added[t] = (pH_min - pH_t[t]) * 2  # Add NaOH to correct pH
        pH_t[t] = pH_min  # Restore pH

# Find Optimal Fermentation Time (t_opt) for Peak Biomass
t_opt = np.argmax(X_t)

# Display Results
print(f'Calculated Reactor Volume: {V * 1000:.3f} mL')
print(f'Optimal Fermentation Time: {t_opt} hours')
print(f'Final Oxygen Flow Rate: {F_O2_t[-1] * 60000:.6f} L/min')
print(f'Total NaOH Added: {np.sum(NaOH_added):.3f} g/L')

# Plot Results
plt.figure(figsize=(12, 8))

# Glucose Depletion
plt.subplot(3,2,1)
plt.plot(time, S_t, 'b', linewidth=2)
plt.xlabel('Time (hours)')
plt.ylabel('Glucose (g/L)')
plt.title('Glucose Depletion Over Time')
plt.grid()

# Biomass Growth
plt.subplot(3,2,2)
plt.plot(time, X_t, 'r', linewidth=2)
plt.xlabel('Time (hours)')
plt.ylabel('Biomass (g/L)')
plt.title('Biomass Growth Over Time')
plt.grid()

# pH Variation
plt.subplot(3,2,3)
plt.plot(time, pH_t, 'g', linewidth=2)
plt.xlabel('Time (hours)')
plt.ylabel('pH')
plt.title('pH Variation Over Time')
plt.grid()

# NaOH Addition
plt.subplot(3,2,4)
plt.plot(time, NaOH_added, 'm', linewidth=2)
plt.xlabel('Time (hours)')
plt.ylabel('NaOH Added (g/L)')
plt.title('pH Correction with NaOH')
plt.grid()

# Oxygen Flow Rate
plt.subplot(3,2,5)
plt.plot(time, F_O2_t * 60000, 'k', linewidth=2)
plt.xlabel('Time (hours)')
plt.ylabel('Oxygen Flow (L/min)')
plt.title('Dynamic Oxygen Flow Rate')
plt.grid()

# CO2 Evolution
plt.subplot(3,2,6)
plt.plot(time, CO2_t, 'c', linewidth=2)
plt.xlabel('Time (hours)')
```

```
plt.ylabel('CO2 (g/L)')
plt.title('CO2 Evolution Over Time')
plt.grid()

plt.tight_layout()
plt.show()
```

 # Google Colab Python Program for Fermentation with pH, Temperature & Foam Control

```
import numpy as np
import matplotlib.pyplot as plt

# Given Parameters
Y_O2_X = 1.1  # Oxygen yield coefficient on biomass (g O2/g cells)
Y_X_S = 0.5   # Biomass yield on glucose (g cells/g glucose)
r_S = 2       # Glucose consumption rate (g glucose/L·h)
k_La = 50     # Oxygen mass transfer coefficient (1/h)
epsilon_g = 0.15  # Gas holdup fraction (15%)
P_O2 = 0.21   # Partial pressure of oxygen in air (atm)
H = 769.2     # Henry's constant for oxygen (atm·L/mol)
C_O2 = 0.005  # Dissolved oxygen concentration (g/L)
C_O2_min = 0.002  # Minimum oxygen level

# Ethanol and CO2 Production
Y_P_S = 0.05  # Ethanol yield on glucose (g ethanol/g glucose)
Y_CO2_S = 1.0 # CO2 yield on glucose (g CO2/g glucose)

# pH Control Parameters
pH_0 = 5.5   # Initial pH
pH_min = 5.0 # Minimum acceptable pH
k_CO2 = 0.05 # CO2 influence on pH drop (empirical factor)
NaOH_added = np.zeros(49)  # Buffer addition tracking (g/L)

# Temperature Control Parameters
Y_Q_X = 15   # Heat yield (kJ/g biomass)
Cp = 4.18    # Specific heat of water (kJ/kg·K)
m = 0.5      # Reactor liquid mass (kg)
T_0 = 30     # Initial temperature (°C)
T_max = 35   # Maximum allowable temperature
Cooling_applied = np.zeros(49)  # Cooling tracking (kJ removed)

# Foam Control Parameters
k_f = 0.02   # Foam factor (L/g gas)
Foam_threshold = 0.1  # Max foam height fraction (10% of reactor height)
AntiFoam_added = np.zeros(49)  # Anti-foam tracking (mL added)

# Solve for initial oxygen flow rate (F_O2)
F_O2 = 0.001785 / 60000  # Convert L/min to m³/s

# Reactor Volume Calculation
V = (F_O2 * Y_O2_X * Y_X_S * r_S) / (k_La * (1 - epsilon_g) * (P_O2 / H - C_O2))

# Time Scale for Fermentation
time = np.arange(49)  # 0 to 48 hours

# Initial Conditions
S0 = 50  # Initial glucose concentration (g/L)
X0 = 0.1  # Initial biomass concentration (g/L)
mu = 0.3  # Specific growth rate (h^-1)

# Preallocate Arrays
S_t = np.zeros_like(time, dtype=float)
```

```
X_t = np.zeros_like(time, dtype=float)
OTR_t = np.zeros_like(time, dtype=float)
OUR_t = np.zeros_like(time, dtype=float)
F_O2_t = np.zeros_like(time, dtype=float)
pH_t = np.zeros_like(time, dtype=float)
CO2_t = np.zeros_like(time, dtype=float)
Temp_t = np.zeros_like(time, dtype=float)
Foam_t = np.zeros_like(time, dtype=float)

# Set Initial Values
S_t[0] = S0
X_t[0] = X0
pH_t[0] = pH_0
Temp_t[0] = T_0
F_O2_t[0] = F_O2

# Loop for Dynamic Control of Aeration, pH, Temperature, and Foam
for t in range(1, len(time)):
    # Biomass Growth
    X_t[t] = X_t[t-1] * np.exp(mu * 1)

    # Glucose Consumption
    S_t[t] = S_t[t-1] * np.exp(-r_S * 1)

    # CO2 Evolution
    CO2_t[t] = Y_CO2_S * (S0 - S_t[t])

    # Oxygen Uptake Rate (OUR)
    OUR_t[t] = Y_O2_X * X_t[t]

    # Oxygen Transfer Rate (OTR)
    OTR_t[t] = k_La * (P_O2 / H - C_O2)

    # Dynamic Oxygen Flow Rate Adjustment
    if OUR_t[t] > OTR_t[t]:
        F_O2_t[t] = F_O2_t[t-1] * 1.1
    elif C_O2 < C_O2_min:
        F_O2_t[t] = F_O2_t[t-1] * 1.2
    else:
        F_O2_t[t] = F_O2_t[t-1]

    # Prevent Excessive Aeration
    if F_O2_t[t] > 0.01:
        F_O2_t[t] = 0.01

    # pH Drop & Control
    pH_t[t] = pH_0 - k_CO2 * CO2_t[t]
    if pH_t[t] < pH_min:
        NaOH_added[t] = (pH_min - pH_t[t]) * 2
        pH_t[t] = pH_min

    # Temperature Increase & Cooling Control
    Heat_production = Y_Q_X * X_t[t]
    Temp_t[t] = T_0 + Heat_production / (Cp * m)

    if Temp_t[t] > T_max:
        Cooling_applied[t] = (Temp_t[t] - T_max) * Cp * m
        Temp_t[t] = T_max

    # Foam Formation & Control
    Foam_t[t] = k_f * (CO2_t[t] + Y_P_S * (S0 - S_t[t]))
    if Foam_t[t] > Foam_threshold * V:
```

```
        AntiFoam_added[t] = 1
        Foam_t[t] = Foam_threshold * V

# Display Results
print(f'Final Temperature: {Temp_t[-1]:.2f} °C')
print(f'Total Cooling Applied: {np.sum(Cooling_applied):.2f} kJ')
print(f'Total Anti-Foam Added: {np.sum(AntiFoam_added):.2f} mL')

# Plot Temperature and Foam Control
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.plot(time, Temp_t, 'r', linewidth=2)
plt.xlabel('Time (hours)')
plt.ylabel('Temperature (°C)')
plt.title('Temperature Control')

plt.subplot(2, 1, 2)
plt.plot(time, Foam_t, 'b', linewidth=2)
plt.xlabel('Time (hours)')
plt.ylabel('Foam Volume (L)')
plt.title('Foam Control')

plt.tight_layout()
plt.show()
```

 # Google Colab Python Program for Sparger Design in a Bubble Column Bioreactor

```
import numpy as np

# Given Parameters
Q_g = 2.2 / 3600  # Gas flow rate (m^3/s) [Converted from L/h]
P_drop = 14000    # Pressure drop across sparger (Pa)
rho_g = 1.225     # Density of air (kg/m^3)
Cd = 0.6          # Discharge coefficient for orifices

desired_velocity = np.sqrt(2 * P_drop / rho_g)  # Gas velocity through orifices (m/s)

# Number of Orifices
N = 6  # Assumed number of orifices

d_o = np.sqrt((4 * Q_g) / (np.pi * N * desired_velocity))  # Orifice diameter (m)

# Bubble Size Estimation
g = 9.81          # Gravitational acceleration (m/s^2)
sigma = 0.072     # Surface tension of water (N/m)
rho_l = 1000      # Density of liquid (kg/m^3)
d_b = 1.38 * ((sigma / (rho_l * g))**(1/3))  # Bubble diameter (m)

# Bubble Rise Velocity using Stokes' Law
mu = 0.001        # Water viscosity (Pa.s)
v_b = (2/9) * ((rho_l - rho_g) * g * d_b**2) / mu  # Bubble rise velocity (m/s)

# Display Results
print('Sparger Design Parameters:')
print(f'Orifice Diameter: {d_o * 1000:.3f} mm')
print(f'Bubble Diameter: {d_b * 1000:.3f} mm')
print(f'Bubble Rise Velocity: {v_b:.3f} m/s')
print(f'Number of Orifices: {N}')
print(f'Gas Velocity Through Orifices: {desired_velocity:.3f} m/s')
```

Table S3 Specification of Auxiliary unit used for the Fabrication of the Bubble Column Bioreactor

S/No	Item name	Picture	Description	Specification
1	**1 Inch Turbine Digital Flow Meter Oval Gear Flow**		The features of the display are, 2 accumulations, resetting and accumulating and optional unit conversion of Gallon and Liter. the high accuracy meter comes with 4 unit options Liters (L) /Gallons (GAL)/ Pints (PT)/ Quarts (QT). The battery inside can supply 9000hours operation approximately. Practical medium: , Liquids	Connector Size: 25mm/ 1inch Max. Working Pressure: 20BAR Working Temp. : 20℃ Max. Flow: 100L/ MIN Min. Flow: 10L/ MIN Precision: ±0.5% Material: Metal Size: Approx. 5.5x 10x 6cm/ 2.16x 3.94x 2.36inch
2	**Digital Pressure Gauge 200 PSI Quick-Connected Air Hose**		The digital pressure gauge measures a range from 0-200 PSI (0-17.2 Bar; 0-1724 Kpa; 0-17.5 kg/cm²) The LED backlit screen features measurements in PSI, KPA, Bar, and Kg/cm²	Material:Heavy Duty Stainless Steel +Brass components Size: 70mmx70mmx35mm Measure Scope: 0-200PSI=13.79(bar)=14.06(kgf/cm2)=1378.95KPa The way of display: Digital LED light Interior Battery: 2xAAA Batteries Apply scope: Universal type Working temperate: -10-50℃ Measurement Units: PSI, KPA, Bar, Kgf/cm2 Thread specification 13mm WITH LCD light
3	**Mini DC 12V Electric Centrifugal Pump Low Noise**		Mini 1/2 Inch DC 12V Electric Centrifugal Water Pump Low Noise 100% new and High Quality This is new pump P-38E, high quality and reliable. Suitable for :, electronic refrigerator, water heater etc.	Rated voltage : 12V DC Current : 500mA (max) Capacity : 6.5L/Min(109 GPH) Pump head : 2M Size : Inner diameter : 1/2 inch(about 12.7mm) Outer diameter : 0.7 inch(about 18mm) But there is about -/+0.03mm difference

4	**Flowmeter Rotameter With Valve Push In 6mm Tube**			PortSize：0.2Mpa Pressure：MediumPressure PowerSource：DC StandardorNonstandard：Standard ActuatorMode：WithoutManualAndInstruction Media：Water IsSmartDevice：YES Power：Pneumatic Material：Other ModelNumber：0.2Mpa
5	**Temperature Controller - 12V**		This is a digital microcomputer temperature controller with led display screen. **Wide temperature** measuring range from -50℃ to 110℃ and temperature control range from -50℃ to 110℃, high accuracy. Easy setting, easy to install and use. Large and clear LCD display for better reading whether it is day or night This advanced device is for controlling temperature. Use in seafood machines, terrarium, vivarium, chicken incubator, etc. Wide temperature control range. Dual heating and cooling function. Temperature measurement function. Large and clear LCD display. Mini and light weight design. Easy to install, simple operation. Temperature control range: - 50 ~ 110°C Temperature measuring accuracy: plus or minus 0.2 °C Temperature control precision: plus or minus 0.1 °C Measuring input: NTC10K Input Voltage: 12V Load power: 120W Appearance size: 60 * 45 * 31 mm This Temperature Controller come with 1 meter waterproof sensor Description: Temperature measurement, LCD display. It has the function of refrigeration, heating control.Temperature setting upper and lower limits.Simple operation, suitable for seafood machines.Easy to install and use, very convenient.Long press the UP button the temperature display flashing, press the UP or DOWN button Set the beginning of the temperature value.Press Down	• LED display • High temperature control precision of 0.1 C • Wide temperature control range from -50 to 110 C • 1m waterproof probe • Convenient and quick

			the display button to stop temperature. Long press the Down button the temperature display flashing, press the UP or Down button Set to stop the temperature value.	
6	**Air Compressor with Pressure Gauge**		**Air Compressor with Pressure Gauge and Three Nozzle Adapters, Portable Metal Cylinder**	**30N/m^2 35L/min Maximum Voltage DC 12V Maximum Amperage Draw 14A.**
7				